Bahan Ajar

PEMROGRAMAN DASAR
Menggunakan Python

Dr. Harja Santana Purba, M.Kom.
Dr. R. Ati Sukmawati, M.Kom.
Muhammad Hifdzi Adini, S.Kom., M.T.

PEMROGRAMAN DASAR
MENGGUNAKAN PYTHON

UU No 28 tahun 2014 tentang Hak Cipta

Fungsi dan sifat hak cipta Pasal 4

Hak Cipta sebagaimana dimaksud dalam Pasal 3 huruf a merupakan hak eksklusif yang terdiri atas hak moral dan hak ekonomi.

Pembatasan Pelindungan Pasal 26

Ketentuan sebagaimana dimaksud dalam Pasal 23, Pasal 24, dan Pasal 25 tidak berlaku terhadap:

i. Penggunaan kutipan singkat Ciptaan dan/atau produk Hak Terkait untuk pelaporan peristiwa aktual yang ditujukan hanya untuk keperluan penyediaan informasi aktual;
ii. Penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk kepentingan penelitian ilmu pengetahuan;
iii. Penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk keperluan pengajaran, kecuali pertunjukan dan Fonogram yang telah dilakukan Pengumuman sebagai bahan ajar; dan
iv. Penggunaan untuk kepentingan pendidikan dan pengembangan ilmu pengetahuan yang memungkinkan suatu Ciptaan dan/atau produk Hak Terkait dapat digunakan tanpa izin Pelaku Pertunjukan, Produser Fonogram, atau Lembaga Penyiaran.

Sanksi Pelanggaran Pasal 113

1. Setiap Orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf i untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan/atau pidana denda paling banyak Rp100.000.000 (seratus juta rupiah).
2. Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf c, huruf d, huruf f, dan/atau huruf h untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan/atau pidana denda paling banyak Rp500.000.000,00 (lima ratus juta rupiah).

Dr. Harja Santana Purba, M. Kom.
Dr. R. Ati Sukmawati, M. Kom.
Muhammad Hifdzi Adini, S. Kom., M.T.

PEMROGRAMAN DASAR
MENGGUNAKAN PYTHON

deepublish
glorify and develop the intellectual of human's life

PEMROGRAMAN DASAR MENGGUNAKAN PYTHON

Harja Santana Purba
R. Ati Sukmawati
Muhammad Hifdzi Adini

Desain Cover :
Nur Anisa Fitriani

Editor:
Mitra Pramita

Tata Letak :
Ika Fatria Iriyanti

Proofreader :
Nuruddin Wiranda

Ukuran :
viii, 103, Uk: 15.5x23 cm

ISBN :
978-623-02-3507-8

Cetakan Pertama :
Oktober 2021

PENERBIT DEEPUBLISH
(Grup Penerbitan CV BUDI UTAMA)
Anggota IKAPI (076/DIY/2012)

Jl.Rajawali, G. Elang 6, No 3, Drono, Sardonoharjo, Ngaglik, Sleman
Jl.Kaliurang Km.9,3 – Yogyakarta 55581
Telp/Faks: (0274) 4533427
Website: www.deepublish.co.id
www.penerbitdeepublish.com
E-mail: cs@deepublish.co.id

KATA PENGANTAR

Syukur alhamdulillaah kami panjatkan ke hadirat Allah subhanahu wata'ala, karena atas ijinnya kami dapat menyelesaikan penyusunan bahan ajar ini. Bahan ajar Pemrograman Dasar dengan Bahasa Python ini dapat digunakan pada mata kuliah Pemrograman Dasar di Program Studi Pendidikan Ilmu Komputer dan mata kuliah Komputer Pemrograman di Program Studi Pendidikan Matematika FKIP Universitas Lambung Mangkurat. Saran dari pembaca akan sangat berguna untuk perbaikan bahan ajar ini. Semoga bahan ajar ini dapat membantu mahasiswa dalam pembelajaran, sehingga dapat meningkatkan hasil belajarnya.

Ucapan terima kasih disampaikan kepada: Dekan FKIP ULM Banjarmasin, Ketua dan Sekretaris Jurusan PMIPA, Ketua dan Sekretaris Prodi Pendidikan Komputer, Reviewer, serta seluruh dosen dan tendik Prodi Pilkom.

September 2021

Tim Penulis

DAFTAR ISI

BAB I PENDAHULUAN

A. Tujuan Pembelajaran

Tujuan yang ingin dicapai dalam pembelajaran ini adalah

1. Mahasiswa dapat membedakan antara algoritma dan program dengan benar.
2. Diberikan suatu masalah, mahasiswa dapat merancang algoritma untuk menyelesaikan masalah tersebut dengan benar dan mempresentasikannya dengan tanggung jawab.
3. Diberikan satu algoritma, mahasiswa dapat menjelaskan langkah kerja dari algoritma dengan benar.

B. Uraian Materi

Satu-satunya keterampilan yang paling penting bagi seorang ilmuwan komputer adalah kemampuan pemecahan masalah. Kemampuan pemecahan masalah berarti kemampuan untuk merumuskan masalah, berpikir kreatif tentang solusi, dan mengungkapkan solusi secara jelas dan akurat (Downey; 2015). Proses belajar membuat program adalah kesempatan yang sangat baik untuk melatih keterampilan memecahkan masalah.

Program adalah urutan instruksi yang menentukan bagaimana melakukan perhitungan/komputasi. Komputasinya mungkin untuk sesuatu yang bersifat matematis, seperti memecahkan sistem persamaan, tetapi bisa juga menjadi komputasi simbolis, seperti pencarian (searching), memproses gambar atau memutar video. Sedangkan programming adalah proses memecahkan tugas yang kompleks dan besar menjadi sub-sub tugas yang lebih kecil sampai sub tugas tersebut cukup sederhana dan dapat dilakukan dengan satu instruksi dasar. Adapun beberapa istilah dasar yang banyak digunakan dalam programming adalah:

Input	: dapatkan data dari keyboard, file, jaringan, atau perangkat lain
Output	: menampilkan data di layar, menyimpannya dalam file, mengirimkannya melalui jaringan, dll.

Math : melakukan operasi dasar matematika seperti penjumlahan dan perkalian

Conditional execution : periksa kondisi tertentu dan jalankan langkah yang sesuai

Repetition : melakukan beberapa tindakan berulang kali

Sebelum dapat menulis program komputer untuk memecahkan masalah, kita harus sudah mengetahui metode untuk menyelesaikan masalah tersebut secara manual. Selanjutnya yang paling penting adalah kita harus dapat mendeskripsikan langkah demi langkah bagaimana masalah tersebut akan diselesaikan. Langkah-langkah yang sistematis inilah yang disebut algoritma. Secara ringkas, tahapan dalam membuat program komputer dapat dilihat pada Gambar 1.1. Jadi membuat program selalu diawali dari deskripsi masalah. Selanjutnya disusun algoritma penyelesaian masalahnya, baru dibuat programnya.

Gambar 1. 1 Alur Pembuatan Program

1.1. Algoritma dalam Pemrograman

Inti yang mendasari semua program komputer adalah algoritma. Algoritma dapat didefinisikan sebagai setiap prosedur komputasi yang terdefinisi dengan baik yang mengambil beberapa nilai, atau kumpulan nilai, sebagai input dan menghasilkan beberapa nilai, atau kumpulan nilai, sebagai output (Cormen, Leiserson, Rivest, dan Stein ; 2009). Menurut (Levitin, 2012), algoritma merupakan urutan instruksi yang tidak ambigu untuk memecahkan masalah dalam jumlah waktu yang terbatas, untuk mendapatkan output yang diperlukan dari setiap input yang sah. Sedangkan menurut (Sedgewick and Wayne; 2011), algoritma adalah metode pemecahan masalah yang terbatas, deterministik, dan efektif yang cocok untuk diimplementasikan sebagai program computer.

Jadi algoritma merupakan prosedur untuk menyelesaikan suatu masalah yang ditulis dengan bahasa sehari-hari ataupun dengan program computer, berupa urutan langkah-langkah komputasi yang tidak ambigu

untuk mengubah input ke output. Sebuah algoritma tidak tergantung pada bahasa pemrograman. Algoritma dapat ditulis dengan bahasa sehari-hari, dengan diagram, atau dengan apa pun yang dapat membantu untuk memvisualisasikan masalahnya. Algoritma kemudian akan diterjemahkan ke dalam Bahasa pemrograman yang dapat diaplikasikan pada computer. Bahasa pemrograman apapun yang digunakan dalam suatu program, selama algoritma yang digunakan sama, maka akan menghasilkan ouput yang sama.

Secara umum setiap algoritma terdiri dari input, proses, dan output seperti tampak pada Gambar 1.2. Seperti diuraikan sebelumnya algoritma merupakan langkah-langkah untuk mengolah input menjadi output. Sehingga suatu algoritma umumnya diawali dengan input, dan diakhiri dengan menghasilkan output.

Gambar 1. 2 Struktur Algoritma

Contohnya, jika kita akan membuat program untuk menghitung luas persegi panjang. Seperti yang telah anda pelajari dalam pelajaran Matematika, luas persegi panjang dapat dihitung dengan cara mengalikan ukuran sisi-sisi persegi panjang yang biasa disebut panjang dan lebar. Sehingga untuk menghitung luas persegi panjang kita perlu input ukuran panjang dan ukuran lebarnya (asumsikan dalam satuan panjang yang sama). Sehingga algoritma untuk menghitung luas persegi panjang diataranya dapat ditulis sebagai berikut:

Contoh 1:
Algoritma LuasPersegiPanjang
1. Masukkan ukuran panjang dan ukuran lebar persegi panjang
2. Hitung luas dengan cara: Luas = panjang ×lebar
3. Tampilkan Luas

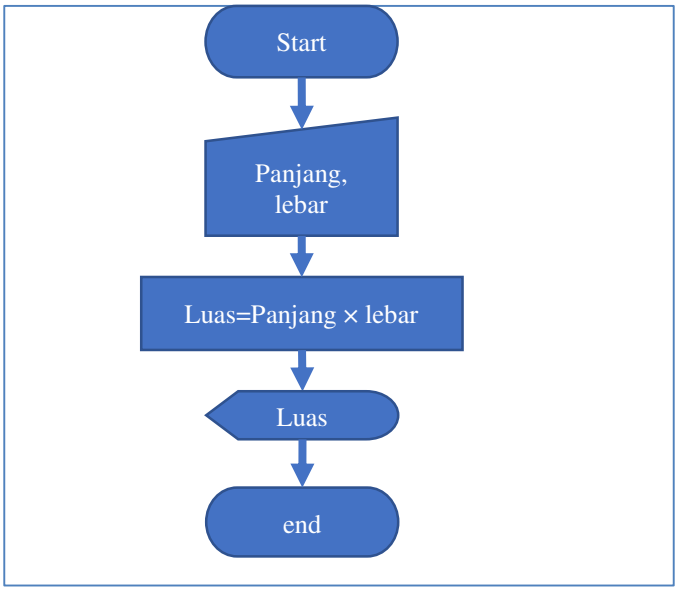
Langkah 1 pada algoritma LuasPersegiPanjang merupakan langkah memasukkan data (input), langkah 2 merupakan langkah untuk memproses

input menjadi output, sedangkan langkah 3 adalah langkah untuk menampilkan output. Saat mempelajari tentang algoritma lebih lanjut, mungkin langkah 3 tidak akan anda temukan. Dalam kuliah ini langkah 3 dituliskan agar saat membuat programnya anda tidak lupa untuk menampilkan output. Algoritma seperti pada contoh 1 merupakan algoritma yang ditulis secara deskriptif.

Algoritma yang ditulis secara deskriptif biasanya lebih jelas tetapi panjang. Sehingga kurang efisien untuk menyelesaikan masalah yang lebih kompleks. Untuk itu dapat ditulis menggunakan pseudocode. Pseudocode merupakan bentuk penulisan algoritma menggunakan kode yang berisi bahasa tiruan dari bahasa pemrograman. Jadi menggunakan symbol-simbol ataupun istilah dalam bahasa pemrograman (biasanya menggunakan Bahasa Inggris), tetapi tidak terikat dengan sintaksnya (aturan-aturan penulisan dalam Bahasa pemrograman). Versi pseudocode dari algoritma pada Contoh 1 diperlihatkan pada Contoh 2.

Contoh 2:
Algoritma LuasPersegiPanjang
1. Input (lebar, panjang)
2. Luas = panjang ×lebar
3. Print (Luas)

Algoritma juga dapat disajikan dalam bentuk diagram alur, atau *flowchart*. Biasanya untuk masalah yang kompleks akan lebih mudah jika dibuat dulu dalam bentuk diagram alur, kemudian dibuat dalam bentuk pseudocode. Gambar 1.3 memperlihatkan diagram alur untuk algoritma menghitung luas persegi panjang.



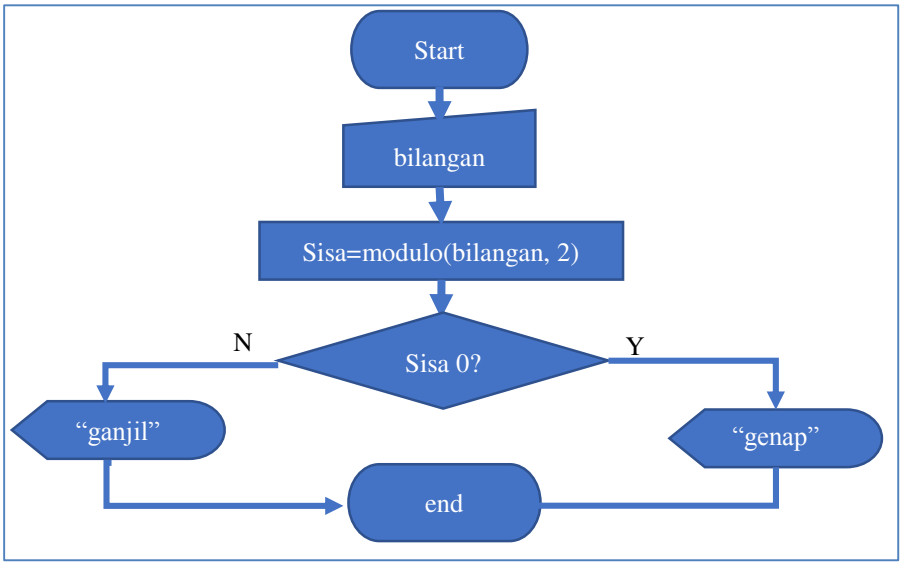
Gambar 1. 3 Diagram Alur Menghitung Luas Persegi Panjang

1.2. Percabangan dan Pengulangan

Penggunaan diagram alur akan sangat membantu ketika menyelesaikan masalah yang berkaitan dengan pengambilan keputusan (*conditional execution*), atau memilih satu dari beberapa alternative yang diberikan. Misalnya akan dibuat algoritma untuk menentukan apakan suatu bilangan bulat merupakan bilangan genap atau ganjil. Seperti diketahui, suatu bilangan bulat disebut bilangan genap jika habis dibagi dua, atau sisa pembagiannya dengan dua adalah nol. Jika tidak habis dibagi dua maka bilangan tersebut merupakan bilangan negative. Alternatif algoritma yang dapat disusun adalah sebagai berikut.

Contoh 3:
Algoritma BilGenapGanjil
1. masukkan (bilangan)
2. hitung sisa pembagian bilangan dengan 2
3. Jika sisa sama dengan 0, maka bilangan genap
   Jika tidak maka bilangan ganjil

Untuk menghitung sisa pembagian dapat digunakan konsep modulo. Sebagai contoh modulo (5,2) adalah 1, yaitu sisa pembagian 5 dengan 2 adalah 1. Demikian juga modulo (8, 3) adalah 2, yaitu sisa pembagian 8 dengan 3 adalah 2. Dengan menggunakan konsep modulo kita dapat menggambar alternatif digram alur untuk menentukan suatu bilangan merupakan bilangan genap atau ganjil, seperti pada Gambar 1.4.

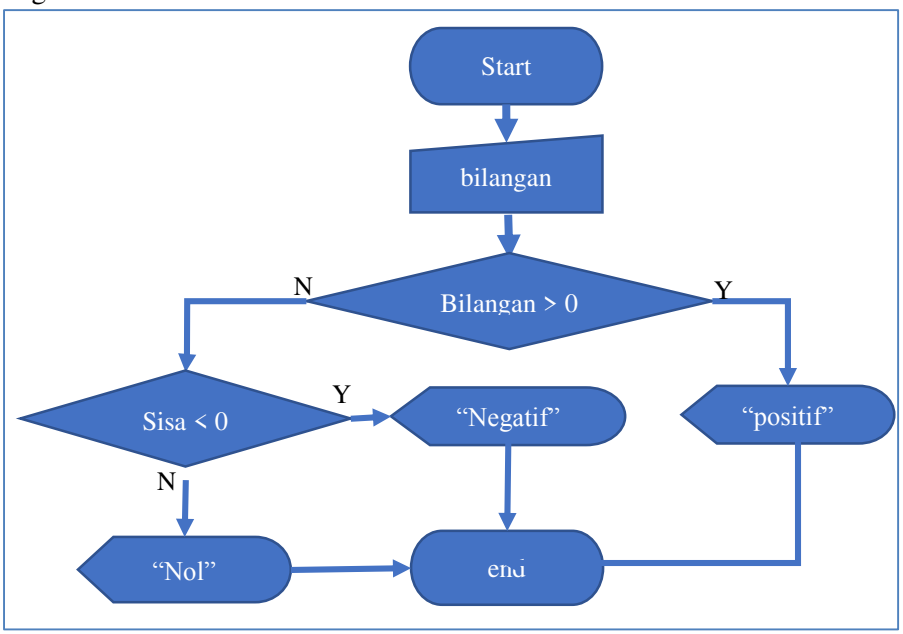


Gambar 1. 4 Diagram Alur Menentukan Bilangan Ganjil atau Genap

Dari Gambar 1.4 dapat kita lihat bahwa suatu bilangan hanya memiliki salah satu sifat, genap atau ganjil. Setelah menghitung modulo, langkah selanjutnya disebut pengujian/pemeriksaan kondisi. Diuji apakah nilai Sisa sama dengan 0. Jika jawaban ya (Y), artinya kondisi dipenuhi maka langkah berikutnya adalah menampilkan pesan bahwa bilangan input merupakan bilangan genap kemudian selesai, jika jawabannya tidak (N), artinya kondisi tidak dipenuhi maka langkah yang dimbil adalah menampilakan pesan bahwa bilangan input merupakan bilangan ganjil kemudian selesai. Jika ditulis dalam bentuk pseudocode diantaranya dapat dilihat pada contoh 4.

Contoh 4:
Algoritma BilGenapGanjil
1. Input (bilangan)
2. sisa = modulo(bilangan, 2)
3. If sisa = 0, then print("bilangan genap")
   Else print ("bilangan ganjil")

Perhatikan diagram alur pada Gambar 1.5. Pada algoritma ini terdapat duakali pengujian nilai bilangan. Algoritma ini digunakan untuk menentukan apakah suatu bilangan merupakan bilangan positif, nol atau negative.



Gambar 1. 5 Diagram Alur Menentukan Bilangan Positif, Negatif, atau Nol

Berdasarkan diagram alur pada Gambar 1.5, dapat dibuat algoritma sebagai berikut.

Contoh 5:
Algoritma BilPositifNegatif

1. Input (bilangan)
2. If (bilangan > 0) then print(“bilangan positif”)
   Else if (bilangan < 0) then print (“bilangan negatif”)
   Else print (“bilangan nol”)

Contoh 4 dan contoh 5 merupakan contoh algoritma untuk kasus menentukan pilihan dari dua dan tiga alternative. Untuk kasus yang lebih kompleks mungkin akan memiliki alternative pilihan yang lebih dari tiga.

1.3. Kesimpulan

Berdasarkan pembahasan pada Bab ini dapat disimpulkan.

1. Program adalah algoritma yang ditulis menggunakan Bahasa pemrograman.
2. Algoritma merupakan prosedur untuk menyelesaikan suatu masalah yang ditulis dengan bahasa sehari-hari ataupun dengan program computer, berupa urutan langkah-langkah komputasi yang tidak ambigu untuk mengubah input ke output.

C. Latihan 1

1. Buat algoritma untuk menghitung luas daerah lingkaran
2. Buat algoritma untuk menentukan luas permukaan balok, jika diketahui ukuran balok
3. Buat algoritma untuk menghitung nilai rata-rata dari tiga bilangan
4. Buat algoritma untuk menentukan z, jika diketahui $z = \frac{2a+b}{ab+1}$
5. Buat algoritma untuk menentukan apakah suatu bilangan bulat merupakan bilangan genap datau bilangan ganjil
6. Buat algoritma untuk menentukan total dari tiga hambatan pada rangkaian listrik
7. Buat algoritma untuk menguji apakah bilangan pertama merupakan factor dari bilangan yang kedua.
8. Buat algoritma untuk mengkonfersi satuan panjang dari centimeter ke satuan panjang lainnya.
9. Sebuah toko menerapkan harga khusus untuk pembelian suatu produk. Jika hanya membeli 1 paket maka harganya 100000 rupiah. Jika membeli 3 paket atau lebih akan mendapat diskon 10% untuk

pembelian ke-2, ke-3 dan seterusnya. Buat algoritma untuk menentukan total harga yang harus dibayar pembeli.
Ilustrasi:
jika membeli 1 paket maka total harga 100000
jika membeli 2 paket maka total harga 200000
Jika membeli 3 paket, maka total harga 100000+90000+90000=280000
Dan seterusnya

10. Buat algoritma untuk menjumlahkan bilangan genap yang kurang dari 50.

BAB 2 PENGENALAN PYTHON

A. Tujuan Pembelajaran

Tujuan yang ingin dicapai setelah pembelajaran ini adalah:

1. Mahasiswa dapat melakukan installasi Python pada computer dengan benar.
2. Mahasiswa dapat membedakan tipe-tipe nilai pada Python dengan benar.
3. Pada shell Python, mahasiswa dapat menggunakan berbagai tipe data, operator, serta variabel untuk membuat ekspresi dalam Python dengan benar.
4. Diberikan suatu deskripsi pernyataan penugasan, mahasiswa dapat menentukan pernyataan penugasan pada Python dengan benar.

B. Uraian Materi

Python merupakan bahasa pemrograman komputer, sama halnya dengan bahasa C, C++, Pascal, Java dan lain-lain. Python disusun oleh Guido Van Rossum pada tahun 1989 di Centrum Wiskunde & Informatica (CWI) Amsterdam Belanda. Publikasi pertama tahun 1991 dengan label versi 0.9.0, yang diikuti dengan versi 1.0 pada tahun 1994.

Setiap bahasa pemrograman memiliki kosakata dan aturan-aturan yang berbeda. Sebagai alternative dari sekian banyak bahasa pemrograman, Python banyak digunakan untuk melakukan administrasi system operasi dan jaringan computer, pengambangan aplikasi desktop maupun web, pengolahan data, dan pembuatan program antar muka ke perangkat keras dan mikrokontroler. Keunggulan Python (Raharjo, 2019):

- Memiliki konsep desain yang bagus dan sederhana. Kode Python mudah dibaca, digunakan ulang dan dirawat.
- Mendukung pemrograman berorientasi objek dan pemrogram funsional
- Untuk memperoleh hasil yang sama, kode Python lebih sedikit dibandingkan dengan kode yang ditulis dalam bahasa lain sehingga dapat menghemat waktu para programmer. Contoh berikut

memperlihatkan perbandingan penulisan kode program dalam Python, C dan C++.

```
# Kode Python
print ("hello world!")
/* Kode C */
#include <stdio.h>
int main(){
    print ("hello world!");
    return 0;
}

/* Kode C++ */
#include <iostream>
int main(){
    std::cout<<"hello world!";
    return 0;
}
```

- Program yang ditulis dalam Python dapat dijalankan di hampir semua system operasi, termasuk untuk perangkat mobile.
- Bersifat gratis dan open source

2.1. Instalasi Python

Langkah pertama untuk memulai pemrograman Python adalah dengan mengunduh terlebih dahulu aplikasi Python pada situs resmi https://www.python.org/downloads/. Python yang digunakan pada modul ini adalah versi 3.8.5. Setelah selesai diunduh, install aplikasi Python pada computer anda, dengan cara double klik pada file python-3.8.5 di Windos Explorer seperti terlihat pada Gambar 2.1. Selanjutnya ikuti instruksi yang ada pada proses installasi sampai selesai. Setelah selesai maka aplikasi siap digunakan.

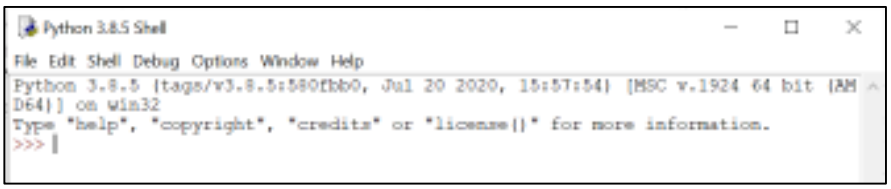
Gambar 2. 1 Instalasi Python

Ketika akan menggunakan Python, cari folder **Python 3.8** pada Start Menu, kemudian pilih **IDLE(Python 3.8 64-bit)** seperti diperlihatkan pada Gambar 2.2.

Gambar2. 2 Membuka Python

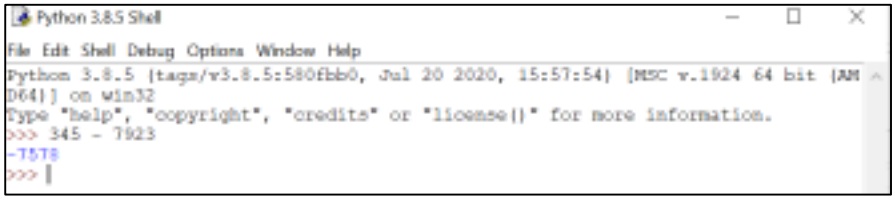
2.2. Python Interpreter

Python Interpreter untuk lingkungan Windows menggunakan program IDLE. Saat dijalankan Python Interpreter akan memunculkan window Python Shell dengan tanda prompt >>>. Pada bagian inilah kita dapat menuliskan perintah-perintah ke dalam interpreter. Gambar 2.3 memperlihatkan tampilan Python Interpreter pada Windows.



Gambar 2. 3 Window Python Shell

Window Python Shell dapat berfungsi seperti kalkulator. Perintah dapat ditulis pada tanda prompt. Misalkan kita akan menghitung hasil pengurangan dari 345 dengan 7923. Tuliskan ekspresi matematisnya 345 – 7923 pada prompt kemudian tekan enter. Maka hasilnya akan Nampak seperti pada Gambar 2.4.



Gambar2. 4 Operasi pengurangan pada Window Python Shell

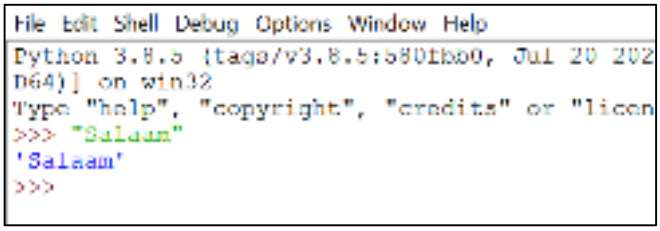
Pada Gambar 2.4 nampak tulisan berwarna biru -7578, yang merupakan hasil pengurangan dari 345 dengan 7923. Pada contoh ini, tulisan

>>> 345 - 7923

merupakan perintah untuk menghitung operasi pengurangan tersebut. Sehingga setelah kita menekan enter, maka baris berikutnya akan muncul hasil perhitungan yang dilakukan oleh mesin. Jadi perintah ditulis pada prompt >>>, hasilnya akan muncul pada baris tanpa prompt >>>. Jika perintah terlalu panjang tidak dapat ditulis dalam satu baris, dapat digunakan tanda *backslash* (\) kemudian enter. Contoh pada Gambar 2.5 memperlihatkan penggunaan tanda *backslash* untuk menulis perintah lebih dari satu baris, yaitu perintah untuk menghitung $2 + 3 - 6 + 5$, ditulis dalam dua baris. Sedangkan baris ketiga merupakan hasil perhitungan yang dilakukan computer, yaitu 4.

Gambar2. 5 Contoh penggunaan lambang backslash

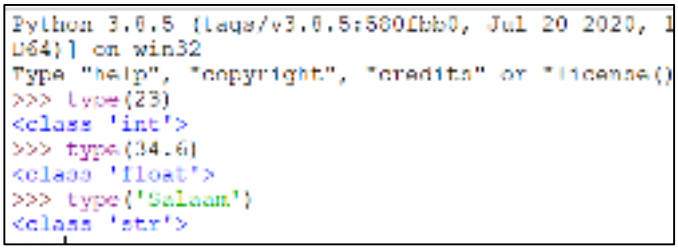
Selain menampilkan bilangan, Python juga dapat menampilkan huruf dan lambang lainnya. Gambar 2.6 memperlihatkan contoh menampilkan kata Salaam.



Gambar2. 6 Menampilkan kata

2.3. Values and Type

Nilai merupakan satu unsur penting yang dikelola oleh program. Nilai dapat berupa bilangan atau huruf. Contoh nilai berupa bilangan adalah 23 dan 34,6. Contoh nilai berupa huruf adalah "Salaam". Setiap nilai memiliki tipe yang berbeda: 23 merupakan bilangan **integer** (bulat), 34,6 merupakan bilangan **floating-point** (bilangan decimal), sedangkan 'Salaam' merupakan **string**. Gambar 2.7 memperlihatkan penggunaan fungsi type untuk mengetahui tipe dari suatu nilai. Ekspresi type(23) menghasilkan output <class 'int'>. Artinya 23 merupakan nilai bertipe integer. Ekpresi type(34.6) menghasilkan output <class 'float'>, artinya 34.6 merupakan nilai yang bertipe floating-point. Output yang terakhir menunjukkan bahwa 'Salaam' merupakan nilai bertipe string.



Gambar2. 7 Memeriksa tipe nilai menggunakan fungsi type

Perhatikan bahwa nilai bertipe string dalam Python selalu ditulis di dalam tanda petik dua ataupun satu. Seperti ″Selamat sore″, atau ′Selamat sore′. Sehingga jika kita menuliskan ekspresi

```
>>> type('8.0')
```

Maka outputnya adalah <class 'str'>. Artinya ketika angka ditulis di dalam tanda petik, maka dia menjadi bertipe string. Karenanya tidak dapat digunakan dalam komputasi matematis. Untuk mengubah tipe dari suatu nilai, dapat digunakan fungsi int, float, atau str sesuai keperluan.

Python memperlakukan bilangan dengan berbagai cara, tergantung bagaimana bilangan tersebut digunakan. Jika bilangan yang digunakan memiliki angka di belakang koma, maka dinyatakan sebagai nilai floating-point atau float. Jika tanpa angka di belakang koma dinyatakan sebagai bilangan bulat atau integer (int). Dalam Python, tidak ada batasan ukuran untuk bilangan bulat, sedangkan untuk bilangan float terdapat batasan. Bilangan decimal ditulis dalam format saintifik.

Contoh:

9.0004523e+5	($9{,}0004523 \times 10^{5}$)
1.002365125127834e6	($1.002365125127834 \times 10^{6}$)
3.234e-12	($3{,}234 \times 10^{-12}$)

Karena bilangan decimal mungkin memiliki angka decimal yang banyak, maka Python menyediakan cara untuk menampilkan bilangan decimal dengan angka decimal tertentu. Pada shell Python, cobalah ketik hal berikut, bandingkan hasilnya.

```
>>> 2/3
>>> format(2/3, '.3f')
```

Apa yang dapat anda simpulkan?

2.4. Operator dan Fungsi

Komputer tidak hanya berfungsi untuk menampilkan nilai ke layar. Selih dari itu, computer dapat melakukan manipulasi terhadap nilai, seperti melakukan operasi aritmatika pada dua nilai integer, atau menggabungkan dua string, dan sebagainya. Operator digunakan untuk melakukan operasi pada nilai data. Untuk operasi pada bilangan digunakan operator aritmatika. Kita dapat menggunakan shell Python sebagai kalkulator untuk melakukan operasi terhadap bilangan. Tabel 2.1 memperlihatkan operator aritmatik yang digunakan dalam Python.

Tabel 2. 1 Operator Aritmatik

Operasi	Lambang	Contoh	Hasil
Penambahan	+	`>>> 2 + 3`	5
Pengurangan	-	`>>> 2-4`	-2
Perkalian	*	`>>> 2*5`	10
Pembagian	/	`>>> 2/5`	0.4
Hasil bagi	//	`>>> 13//5`	2
Sisa Pembagian (modulo)	%	`>>> 13%5`	3
Perpangkatan/ eksponen	**	`>>> 3**2` `>>> 27**(1/3)`	9 3.0

Selain operator aritmatik, Python juga menyediakan fungsi-fungsi matematika yang dapat digunakan untuk memanipulasi bilangan. Tabel 2.2 menampilkan beberapa fungsi matematika yang umum digunakan.

Tabel 2. 2 Beberapa fungsi Matematika

Matematika	Python	Contoh	Hasil
x^y	`pow(x, y)`	`>>> pow(2,3)` `>>> pow(27,(1/3))`	8 3.0
`max(x1,x2, …)`	`max(x1,x2, …)`	`>>> max(23, 56, 4)`	56
`min(x1,x2, …)`	`min(x1,x2, …)`	`>>> min(2,5,1,0,-1)`	-1
`|x|`	`abs(x)`	`>>> abs(-4)`	4

Operator lain yang sering digunakan dalam pemrograman adalah operator relasional. Operator ini digunakan untuk mengevaluasi hubungan antara dua nilai numerik. Misalnya pernyataan 2<3, tidak menghasilkan suatu bilangan tetapi menghasilkan nilai benar atau salah. Berbagai operator relasional dapat dilihat pada Tabel 2.3.

Tabel 2. 3 Tabel 2.3

Operator	Keterangan	Contoh	Hasil
==	Sama dengan	`>>> 4==5`	`False`
!=	Tidak sama dengan	`>>> 4!=5`	`True`
>	Lebih dari	`>>> 4>5`	`False`
<	Kurang dari	`>>> 4<5`	`True`
<=	Kurang dari sama dengan	`>>> (4-2)<=6`	`True`
>=	Lebih dari sama dengan	`>>> (4-2)>=6`	`False`

Selain operator relasional, Python juga menyediakan operator logika. Seperti operator relasional, operator logika digunakan untuk mengevaluasi dua nilai sesuai logika matematika. Tabel 2.4 memperlihatkan operator logika dan contoh penggunaannya.

Tabel 2. 4 Operator logika

Operasi	Operator	Keterangan	Contoh	Hasil
Dan	And	Bernilai benar, jika kedua argument bernilai benar	>>> 2<3 and 4>3	True and True, maka True
Atau	Or	Bernilai benar, jika minimal satu argument bernilai benar	>>> 3<2 or 2>3	False or False, False
Negasi	Not	Benar, jika argumen bernilai salah, dan sebaliknya	>>> not(5>2)	Not (True), maka False

2.5. Variabel, Ekspresi dan Penugasan

Saat membuat program akan banyak nilai yang harus disimpan di memori, untuk itu digunakan variabel. Seperti dalam aljabar, variabel berfungsi untuk menyimpan nilai. Dalam pemrograman variabel merupakan ruang dalam memori yang digunakan untuk menyimpan nilai. Agar dapat diakses variabel diberi identitas, yaitu nama variabel. Setiap bahasa pemrograman memiliki aturan masing-masing dalam penamaan variabel. Aturan penamaan variabel dalam Python adalah sebagai berikut:

1) Nama variable hanya memuat huruf, angka dan garis bawah (underscores).
2) Diawali dengan huruf atau garis bawah. Misalnya: Nilai_1. Tidak boleh 1_Nilai
3) Tidak boleh mengandung spasi. Gunakan garis bawah untuk penghubung.
4) Tidak menggunakan keyword ataupun nama fungsi dalam Python, seperti print, system, if, for, dll

5) Python membedakan penggunaan huruf besar dan huruf kecil dalam menamaan variabel, sehingga Nilai dan nilai akan merujuk ke dua variabel yang berbeda.
6) Sebaiknya singkat dan mendeskripsikan data yang disimpan dengan jelas. Misalnya: nama variabel nama_siswa lebih baik dari pada n_s.

Kombinasi dari nilai, variabel, dan operator akan menghasilkan suatu ekspresi. Sebelumnya kita sudah membuat berbagai ekspresi pada Python yang memuat bilangan dan operator, seperti 5+7, 6 >=9, dan sebagainya. Berikut adalah contoh ekspresi yang merupakan gabungan variabel (k), operator (-) dan nilai (3).

```
>>> k -3
```

Akan tetapi jika ekspresi ini dijalankan pada Shell Python, maka akan muncul pesan kesalahan. Ini terjadi karena variabel k belum didefinisikan. Jadi sebelum digunakan, variabel harus didefinisikan/ diset terlebih dahulu. Untuk keperluan ini digunakan skema penugasan atau *assignment*.

Penugasan digunakan untuk membuat dan mengisi nilai dari variabel. Misalnya untuk membuat variabel x, dan isi dengan nilai 5. Dalam aljabar biasa ditulis x = 5. Dalam Python berlaku hal yang sama, kita tulis

```
>>> x = 5
```

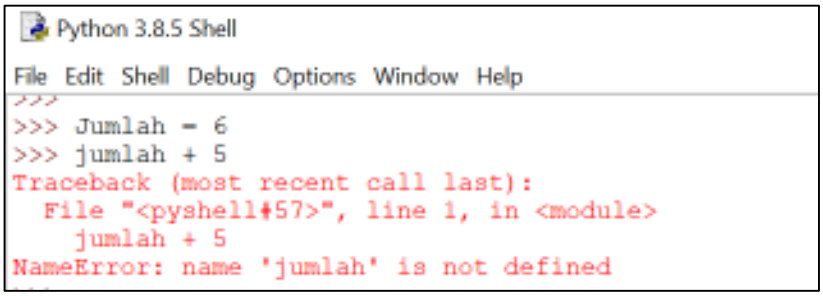
Pernyataan x = 5, disebut pernyataan penugasan (assignment). Artinya isi variable x dengan 5. Bentuk umum pernyataan penugasan adalah

<variable> = <ekspresi>

Gambar 2.8 memperlihatkan contoh penugasan, yaitu Jumlah = 12. Karena baris sebelumnya tidak ada variabel Jumlah, maka dengan penugasan ini dibentuk variabel Jumlah dan mengisinya dengan nilai 12. Ketika ditekan enter, Python tidak akan menampilkan apapun. Untuk melihat nilai dari variabel Jumlah hasil penugasan tersebut, ditulis pernyataan pada baris kedua. Pengetikan nama variabel pada promt shell merupakan perintah untuk menampilkan nilai yang tersimpan pada variabel Jumlah, yaitu 12. Pernyataan Jumlah+3, merupakan ekspresi untuk menambahkan 3 ke nilai dari variabel Jumlah dan menampilkan hasilnya, yaitu 15. Jadi penulisan ekspresi dan penugasan pada Shell Python memberikan efek yang berbeda.

Gambar2. 8 Contoh ekspresi dan penugasan

Ketika kesalahan penulisan variabel muncul pada program, program IDLE akan menampilkan pesan dimana letak kesalahan yang dilakukan. IDLE menyediakan suatu traceback untuk mengetahui letak kesalahan, seperti diperlihatkan pada Gambar 2.9.



Gambar2. 9 Traceback untuk mengetahui letak kesalahan

Tulisan merah pada Gambar 2.9 menyampaikan pesan bahwa kesalahan terletak pada pernyataan jumlah+5. Nama kesalahannya adalah nama variabel ‘jumlah’ tidak dikenal (name ‘jumlah’ is not defined). Jika kita perhatikan, pada baris pertama terdapat pernyataan penugasan Jumlah=6. Jadi dalam program ini terdapat kesalahan dalam penulisan nama variabel. Variabel yang ada adalah Jumlah, sedangkan variabel jumlah tidak dikenal. Kesalahan nama variable bisa terjadi karena berbagai factor. Bisa karena nilai dari variable tersebut belum diseting/ didefinisikan sebelumnya, atau ada kesalahan pada penulisan (penggunaan huruf besar dan kecil, atau kehilangan satu huruf dll).

Setiap variabel setiap saat hanya berisi satu nilai. Ketika nilai yang baru masuk, maka nilai yang lama akan terhapus. Perhatikan penugasan berikut.

```
>>> x = 5
>>> x = x + 3
>>> y = x
>>> y = y - 5
```

Pada penugasan pertama variabel x dibuat dan diisi dengan 5. Selanjutnya pada penugasan kedua variabel x akan diisi dengan penjumlahan nilai variabel x yang lama dengan 3. Sehingga variabel x setelah penugasan yang kedua akan berisi nilai 8. Berapakah nilai dari variabel x dan y di akhir penugasan?

2.6. Kesimpulan

1. Nilai dapat berupa bilangan atau huruf.
2. Tipe Nilai yang dapat digunakan terdiri dari integer (int), floating-point (float), dan **string**.
3. variabel merupakan ruang dalam memori yang digunakan untuk menyimpan nilai.
4. Aturan penamaan variabel dalam Python adalah sebagai berikut:
   a. Nama variable hanya memuat huruf, angka dan garis bawah (underscores).
   b. Diawali dengan huruf atau garis bawah.
   c. Tidak menggunakan keyword ataupun nama fungsi dalam Python, seperti print, system, if, for, dll
   d. Python membedakan penggunaan huruf besar dan huruf kecil dalam menamaan variabel.
5. Ekspresi adalah kombinasi dari nilai, variabel, dan operator.
6. Penugasan adalah pernyataan untuk membuat dan mengisi nilai dari variabel, atau perintah untuk melakukan sesuatu.

C. Latihan 2

1. Buatlah ekspresi aritmatik pada Shell Python untuk menentukan
   a. Jumlah dari lima bilangan ganjil pertama
   b. Rata-rata dari nilai ujian Fatimah, Ahmad dan Budi, jika nilai Fatimah 67, nilai Ahmad 85, dan nilai Budi 94.

c. $\frac{3^2-\sqrt[3]{64}}{5+\sqrt{9}}$
d. Sisa pembagian 435 dengan 74
e. Nilai terendah dari 45, 89, 56, 38, 69

2. Pada shell Python buat ekspresi untuk mengetahui hasil dari pernyataan-pernyataan berikut
   a. Jumlah dari 2 dengan 9 lebih dari sama dengan 9
   b. Hasil dari 7//3 tidak sama dengan hasil dari 8-6
   c. Jumlah dari 6 kuadrat dengan 8 kuadrat sama dengan 10 kuadrat
   d. Jumlah dari 4, 6 dan 8 kurang dari 18
   e. 4324 habis dibagi 12
   f. 62 adalah bilangan ganjil
   g. Bilangan terbesar dari 2,7,3,12, 5 merupakan bilangan genap
   h. Bilangan terkecil dari 87, 54, 90, 45 sama dengan nilai rata-ratanya
3. Tulis pada Shell Python anda
   a. pernyataan penugasan mengisi nilai 3 ke variabel y
   b. Assign 8 ke variabel x
   c. Assign ke variabel z, nilai dari ekspresi $2x + y^3 - 6$
   d. Tampilkan nilai variabel z
   e. Buat ekspresi untuk menentukan nilai terendah dari z-y, dan z-x
   f. Buat penugasan untuk mengisi nilai variabel bilangan, dengan kuadrat dari nilai variabel bilangan yang lama ditambah 3

BAB 3 MEMULAI PROGRAMMING DENGAN PYTHON

A. Tujuan Pembelajaran

Tujuan yang ingin dicapai setelah pembelajaran ini adalah:

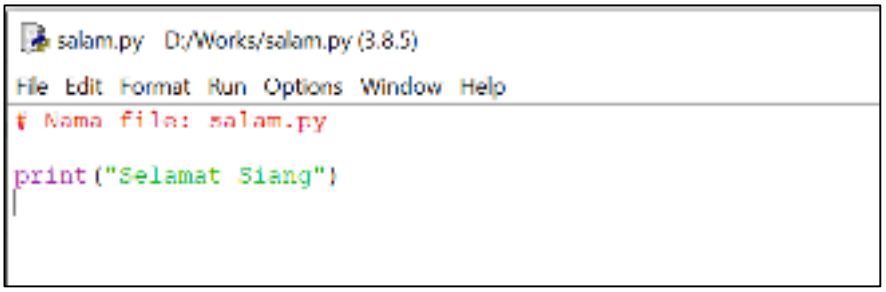
1. Mahasiswa dapat membuat program sederhana, menyimpannya pada file script dan menjalankan program melalui windows explorer dengan benar.
2. Mahasiswa dapat mengimplementasikan penggunaan fungsi print() untuk mengatur tampilan output program dengan kreatif.
3. Mahasiswa dapat mengimplementasikan penggunaan fungsi input() dalam program Python, untuk mengatur inputan data dari pengguna.
4. Diberikan suatu masalah, mahasiswa dapat menghasilkan program interaktif dalam Python untuk menyelesaikan masalah tersebut dengan memanfaatkan module *math*
5. Diberikan suatu masalah, mahasiswa dapat menghasilkan program interaktif dalam Python untuk menyelesaikan masalah tersebut dengan memanfaatkan *fractions*

B. Uraian Materi

Walaupun kita dapat bekerja secara interaktif pada window Python Shell, tetapi menjadi tidak efisien ketika program yang ditulis cukup panjang. Alternatifnya adalah kita menyimpan kode-kode Python ke dalam file script. Untuk menuliskan kode program tersebut digunakan editor Python. Klik tab File-New File. Maka akan terbuka window baru seperti diperlihatkan pada Gambar 3.1, yang merupakan tempat untuk kita menuliskan kode-program.

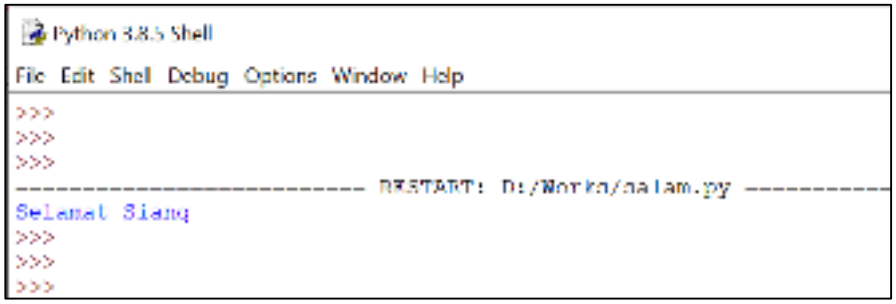
Gambar 3. 1 Window Editor Python

Sebelum membuat kode program, buatlah direktori kerja. Kita akan membuat direktori kerja di D dengan nama Work (D:\Work). Setelah direktori kerja siap, pada window editor ketik ulang program seperti diperlihatkan pada Gambar 3.2, kemudian simpan pada direktori kerja sebagai file salam.py.



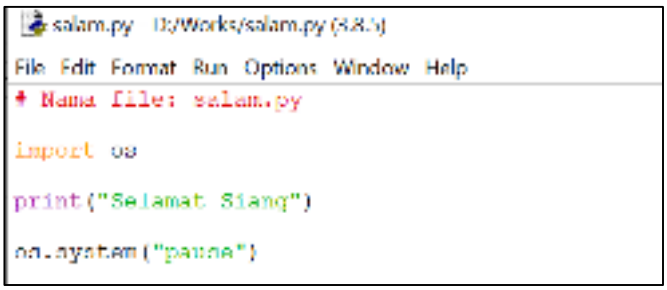
Gambar 3. 2 Tampilan File salam.py

File program kita sudah tersimpan di D:\Works dengan nama file salam.py. Program tersebut akan menampilkan teks Selamat siang ke layar. Untuk menjalankan program klik tab **Run-Run Module (F5)**. Maka hasil program akan ditampilkan pada window Python shell, seperti diperlihatkan pada Gambar 3.3.



Gambar 3. 3 Tampilan output program salam.py

Kita dapat menjalankan file program salam.py langsung melalui direktori kerja kita. Klik-dua kali (double click) atau sorot nama filenya kemudian enter. Program akan dieksekusi, dan ketika selesai jendelanya akan langsung tertutup. Agar program tidak langsung ditutup setelah eksekusi, kita dapat menggunakan fungsi **`system`**`()` dalam modul os untuk megirimkan pesan pause (lihat contoh pada Gambar 3.4).



Gambar 3. 4 Contoh penggunaan modul os

Penggunaan perintah pause pada program, menyebabkan program akan menunggu aksi dari pengguna untuk melanjutkan eksekusi program. Jika kita eksekusi program salam.py langsung dari direktori kerja, maka hasilnya akan tampil seperti pada Gambar 3.5. Ketika pengguna menekan sembarang tombol pada keyboard, maka eksekusi program akan diselesaikan dan windows py.exe akan ditutup.

Gambar 3. 5 Contoh hasil penggunaan modul os

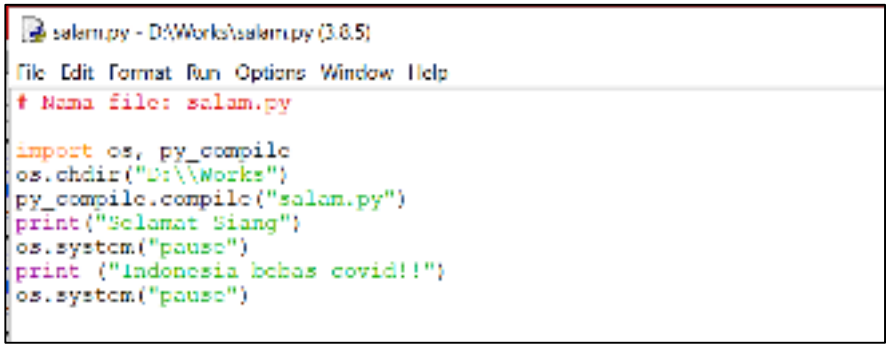
Latihan:

1. Ketik ulang program pada Gambar 3.6 di editor Python, kemudian jalankan langsung melalui direktori kerja. Jelaskan apa yang terjadi!

```
# Nama file: salam.py

import os

print("Selamat Siang")
os.system("pause")
print ("Indonesia bebas covid!!")
os.system("pause")
a = 6
print (a)
print (a+7)
os.system("pause")
```

Gambar 3. 6 Soal latihan 1

2. Perhatikan direktori kerja anda pada windows explorer. File apa saja yang anda lihat pada direktori tersebut?. Ketik ulang program pada Gambar 3.7, kemudian jalankan. Amati file baru apa yang muncul pada direktori D:\\Works anda. Apa perbedaan dan persamaan file baru tersebut dengan file salam.py pada Gambar 3.7?



Gambar 3. 7 Soal latihan 2

3.1. Fungsi print

Output dari suatu program dapat pada umumnya akan ditampilkan ke layar komputer. Pada baris ketiga dari program pada Gambar 3.2 tertulis

```
print("Selamat Siang")
```

Baris ini merupakan suatu perintah atau penugasan. Fungsi **print** pada baris ini merupakan fungsi untuk menampilkan sesuatu ke layar, dalam hal ini menampilkan semua karakter yang ada dalam dua tanda petik, yaitu kalimat Selamat Siang. Dengan menggunakan fungsi **print**, komputer ditugaskan untuk menampilkan ke layar komputer semua karakter yang ditulis di dalam tanda petik-dua ataupun tanda petik-satu.

Pada Bab sebelumnya telah dibahas cara menampilkan nilai variabel pada Python Shell, yaitu dengan menuliskan nama variabel pada prompt seperti berikut

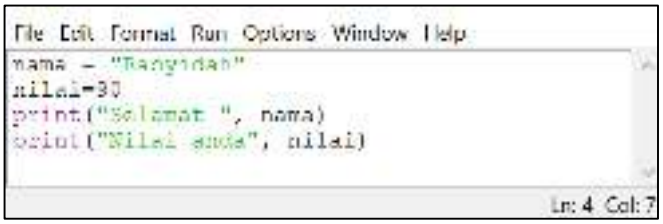
```
>>> Jumlah = 9 +3
>>> Jumlah
```

Apa yang terjadi jika hal yang sama dilakukan pada mode script? Dalam mode script hasilnya akan berbeda. Untuk menampilkan nilai dari suatu variabel dalam mode script harus digunakan fungsi **print**(). Misalkan akan menampilkan nilai dari variabel Luas, maka dapat dilakukan dengan perintah penugasan

```
print(Luas)
```

Pada Gambar 3.6 terdapat perintah **`print(a+7).`** Pada kasus ini, a+7 tidak ditulis di dalam tanda petik. Artinya a+7 bukan nilai bertipe string tetapi bilangan. Dalam hal ini karena variabel a bernilai 6, maka akan ditampilkan ke layar nilai 13.

Ketika ingin menampilkan string dan nilai variabel, dapat dilakukan dengan menggunakan pemisah tanda koma. Contohnya dapat dilihat pada Gambar 3.8. Variabel nama bertipe string, sedangkan variabel nilai bertipe integer.



Gambar 3. 8 Menampilkan string dan nilai variabel

Default dari fungsi print setelah selesai menampilkan objek ke layar maka kursor akan pindah ke baris baru. Sehingga perintah print berikutnya akan menampilkan objek pada baris baru. Output dari program 3.8 adalah

Selamat Rasyidah
Nilai anda 90

Jika ingin menampilkan dalam satu baris, maka dapat dilakukan dengan penggunaan satu fungsi print ataupun tetap menggunakan dua fungsi print atau dapat juga dengan menggunakan satu fungsi print dan fungsi format. Coba dan pelajarilah program pada Gambar 3.9.

```
File Edit Format Run Options Window Help
nama = "Rasyidah"
nilai=90
print("Menggunakan dua fungsi print")
print("Selamat", nama, end=' ')
print("Nilai anda", nilai)
print()
print('Menggunakan satu fungsi print')
print("Selamat", nama, "Nilai anda", nilai)
print()
print('Menggunakan fungsi format()')
print("Selamat {} Nilai anda {}".format(nama, nilai))
```

Gambar 3. 9 Menampilkan objek ke layar dalam satu baris

Seperti dikemukakan sebelumnya, untuk bilangan decimal Python dapat menampilkannya dalam beberapa angka di belakang koma, yaitu menggunakan format. Misalkan akan menampilkan nilai dari variabel bilangan dalam 2 angka di belakang koma adalah sebagai berikut

```
print(”bilangan = %.2f” % bilangan)
print(f'{bilangan:.2f}')
```

atau menampilkan dalam format bilangan eksponen. Misalnya jika x=2600000 = $2{,}6.10^6$ = 2,6e+06. Untuk ini dapat digunakan pernyataan berikut

```
print(f'{x:.2e}')     #hasil 2.60e+06
print(f'{x:.1e}')     #hasil 2.6e+06
```

Mengatur tampilan output dapat juga dilakukan dengan memanfaatkan karakter escape pada Python. Karakter escape merupakan karakter yang ditulis setelah tanda *backslash* (\) dan memiliki fungsi tertentu sesuai dengan kode karakter masing-masing. Tabel 3.1 memperlihatkan beberapa karakter escape yang dapat dipakai dalam pengaturan tampilan output dalam Python.

Tabel 3. 1Karakter Escape

Notasi	Deskripsi
\a	Peringatan
\b	Backspace
\n	Newline

\t	Tab
\u	Format pencetakkan (termasuk symbol), misal: **print("x\u00b2") untuk x^2** **print("a\u2081") untuk a_1** **print("x\u00b0") untuk x°** **print("\u03A0") untuk Π**

3.2. Komentar

Dalam suatu program selain kode program, biasa terdapat penjelasan-penjelasan tentang program tersebut. Contohnya pada program Gambar 3.2. di baris pertama tertulis

```
# Nama File: salam.py
```

Ini merupakan suatu pemberitahuan bahwa file program tersebut diberi nama salam.py. Semua karakter yang ditulis setelah lambang #, sampai akhir baris, menunjukkan bahwa baris tersebut merupakan keterangan. Interpreter dari Python tidak menganggap baris itu ada (sebagai komentar saja). Jika komentar lebih dari satu baris dapat ditulis dengan tanda # pada setiap awal baris, seperti diperlihatkan pada Gambar 3.10.

Gambar 3. 10 Contoh penulisan komentar lebih dari satu baris

Komentar yang lebih dari satu baris dapat juga ditulis di dalam tanda petik, seperti diperlihatkan pada Gambar 3.10. Menempatkan komentar tidak harus di awal baris, boleh juga ditempatkan di belakang setelah pernyataan.

Gambar 3. 11 Penulisan komentar menggunakan tanda petik

Latihan:

1. Buatlah kode program dalam bahasa Python. Program akan menampilkan biodata dan deskripsi diri anda. Program minimal terdiri dari 50 baris.
2. Buatlah kode program dalam bahasa Python, untuk menampilkan inisial nama anda pada seluruh layar secara kreatif.

3.3. Fungsi input

Seperti telah dibahas sebelumnya, pengisian nilai variabel pada Python dapat dilakukan melalui pernyataan penugasan. Pada Bab sebelumnya telah dibahas mengenai algoritma untuk menghitung luas persegi panjang, yaitu:

1. Input panjang, lebar
2. Luas = panjang ×lebar
3. Tampilkan Luas

Langkah 1 dari algoritma ini dapat dimaknai, mengisi variabel panjang, dan mengisi variabel lebar. Hal ini dapat dilakukan dengan cara penugasan langsung pada kode program, seperti berikut.

```
panjang = 6
lebar = 5
```

Artinya variabel panjang diberi nilai 6, danvariabel lebar diberi nilai 5. Langkah 2 juga merupakan pengisian nilai variabel. Yaitu mengisi nilai variabel Luas dengan hasil perkalian nilai variabel panjang dengan nilai variabel lebar. Sehingga dapat ditulis dalam bentuk penugasan

```
Luas = panjang * lebar
```

Jika program ini kita jalankan, maka nilai-nilai variabel panjang dan lebar dari waktu ke waktu bersifat tetap, yaitu 6 dan 5. Berapa kalipun program dijalankan, selama kode programnya tidak diubah maka nilainya tetap. Agar nilai dari suatu variabel dapat bervariasi sesuai keperluan, maka diperlukan masukkan nilai variabel dari pengguna. Lalu bagaimana caranya mengisi nilai variabel oleh pengguna?

Dalam Python untuk keperluan pemasukkan data dari pengguna dapat dilakukan dengan memanfaatkan fungsi **`input`**. Gambar 3.12

memperlihatkan contoh program penggunaan fungsi **input** untuk mengisi nilai dari variabel Nama.

```
File Edit Format Run Options Window Help
Nama = input('Nama anda adalah: ')
print()
print('Selamat datang ', Nama)
print('Selamat belajar programming dengan Python')
```

Gambar 3. 12 Contoh penggunaan fungsi input

Jika program pada Gambar 3.12 dieksekusi, dengan menekan tombol f5, maka pada window shell akan muncul seperti pada Gambar 3.13.

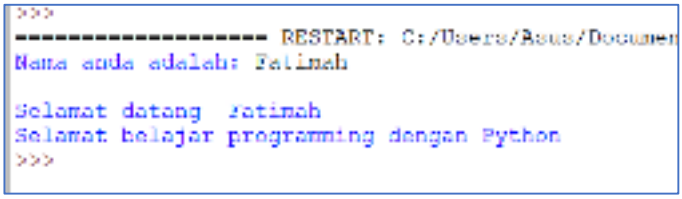
```
                    RESTART: C:/Users
Nama anda adalah:
```

Gambar 3. 13 Contoh hasil eksekusi perintah input

Kalimat pada Gambar 3.13 merupakan tampilan dari hasil eksekusi perintah input pada baris pertama dari program Gambar 3.12, yaitu

```
Nama = input ('Nama anda adalah: ')
```

Pernyataan ini merupakan suatu penugasan, yaitu mengisi nilai dari variabel Nama. Fungsi **input**, mengisaratkan bahwa inputan dilakukan dari pengguna melalui *keyboard*. Ketika dijalankan, pada Gambar 3.13 dapat dilihat bahwa program hanya menampilkan string yang ada di dalam fungsi **input**, yaitu 'Nama anda adalah:', dengan Kursor terletak di belakang tanda titik dua. Ini menunjukkan bahwa program menunggu suatu masukkan dari pengguna. Ketika kita ketikan satu string, misalnya Fatimah, kemudian tekan enter, maka program akan menjalankan perintah berikutnya. Baris kedua adalah penugasan untuk pindah ke baris kerikutnya, sedangkan baris ketiga merupakan penugasan untuk menampilkan string 'Selamat datang' diikuti dengan nilai dari variabel Nama. Hasil eksekusi selengkapnya dapat dilihat pada Gambar 3.14.

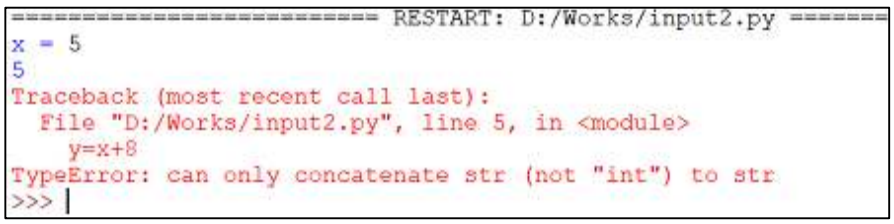


Gambar 3. 14 Hasil eksekusi selengkapnya program pada Gambar 3.12

Fungsi **input** pada Python selalu menghasilkan suatu string. Artinya bilangan yang diinputkan akan dibaca sebagai string. Sehingga jika kita memasukkan suatu bilangan, maka bilangan tersebut tidak dapat dioperasikan secara aritmatika. Misalkan kita menuliskan program sebagai berikut:

```
x=input()
print(x)
y=x+8
print(y)
```

Maka akan muncul pesan kesalahan seperti pada gambar 3.15. Pada pesan tersebut tertulis bahwa penugasan **y=x+8** pada program tersebut salah. Pada baris terakhir disebutkan tipe kesalahan yang terjadi, yaitu bahwa nilai sting hanya bisa dikonkatenasi dengan string. Hal ini terjadi karena variabel x dengan perintah **input** akan diisi dengan suatu string. Karena variabel x merupakan string, maka operasi penjumlahan pada string (disebut concaten) merupakan penggabungan dua string. Sedangkan 8 bukan suatu string, sehingga penugasan ini tidak dapat dilakukan.



Gambar 3. 15 Contoh pesan kesalahan dalam penggunaan fungsi input

Karena fungsi **input** selalu menghasilkan suatu string, untuk mengubahnya menjadi bilangan dapat digunakan fungsi **int ()** atau **float ()** atau sesuai keperluan**.** Jadi jika ditulis

```
panjang = input()
```

maka nilai dari variabel panjang adalah sebuah string. Sehingga tidak dapat dioperasikan sebagai bilangan. Tetapi jika ditulis,

```
panjang = int (input())
```

maka variabel panjang dapat diisi dengan nilai integer dan dapat diperlakukan sebagai bilangan. Gambar 3.16 memperlihatkan contoh penggunaan fungsi **input** dan **print** pada program untuk menentukan luas permukaan balok. Selain fungsi **int** dan **float** mengubah masukkan dari string ke bilangan dapat juga menggunakan fungsi **eval**().

```
#Program menghitung luas persegi panjang

Pjg = int(input('panjang='))
Lbr = int(input('lebar ='))
Luas = Pjg*Lbr
print('luas persegi panjang dengan panjang ', Pjg,' dan lebar ', Lbr, ' adalah ', Luas)
```

Gambar 3. 16 Contoh program interaktif menggunakan fungsi input

Jika program pada Gambar 3.16 dieksekusi, kemudian panjang diisi dengan 23, dan lebar diisi dengan 3. Maka hasil eksekusi selengkapnya dapat dilihat pada Gambar 3.17.

```
panjang=23
lebar =3
69
>>>
```

Gambar 3. 17 Hasil eksekusi selengkapnya program LuasBalok

Latihan:

1. Buat program untuk menghitung rata-rata dari tiga nilai ulangan, dengan tiga nilai ulangan merupakan inputan dari pengguna.
2. Buat program yang meminta pengguna untuk memasukkan empat bilangan bulat. Selanjutnya program akan menampilkan nilai

kebenaran dari pernyataan nilai rata-rata dari keempat bilangan yang diinputkan kurang dari 75.

3. Buat program untuk mempertukarkan nilai dari dua variabel yang merupakan inputan dari pengguna.
4. Buat program yang meminta pengguna memasukkan lima bilangan bulat. Selanjutnya program akan menampilkan bilangan terbesar dan bilangan terkecil dari bilangan yang diinputkan.
5. Buat program untuk menghitung nilai dari x^2+x+1, dengan nilai x merupakan inputan dari pengguna. Gunakan karakter escape yang sesuai untuk menampilkan luaran program.

3.4. Module math

File salam.py yang telah dibuat dan disimpan sebelumnya merupakan suatu contoh module yang didefinisikan oleh pemakai (*user-defined*) dalam Python. Suatu module merupakan file sederhana yang berisi kode Python. Setiap file yang berisi kode Python dan nama filenya berekstensi .py merupakan module Python. Jadi file salam.py merupakan module, demikian juga file math.py, fractions.py, dan Luas.py.

Pada dasarnya bahasa Python hanya mendukung operasi dasar matematika. Untuk fungsi-fungsi matematika seperti fungsi eksponen atau fungsi trigonometri diperlukan module math. Module math merupakan module yang berisi konstanta dan fungsi-fungsi matematis. Untuk menggunakan module ini harus diimport secara eksplisit dengan perintah **`import`**. Gambar 3.18 memperlihatkan contoh penggunaan module math.

```
import math
x = 9
y = math.sqrt(x)
print (y)
print(y*5)
```

Gambar 3. 18 Penggunaan module math

Program pada Gambar 3.18 menggunakan fungsi sqrt() yang terdapat pada module math. Fungsi ini untuk menghitung akar dari suatu bilangan. Selain fungsi sqrt() beberapa fungsi dan konstanta yang ada pada module

math dapat dilihat pada tabel 3.2. Untuk melihat selengkapnya isi dari module math digunakan perintah *help(math).*

Tabel 3. 2 Beberapa fungsi pada module math

Fungsi	Keterangan
sqrt(x)	$\sqrt{x}$
ceil(x)	Pembulatan ke atas
floor(x)	Pembulatan ke bawah
cos (x)	Fungsi cosinus
sin (x)	Fungsi sinus
log(x, n)	$\log_n x$
Pi	Konstanta: 3,141592653589793
E	Konstanta: 2,718281828459045
exp(x)	e^x
pow(x,y)	x^y

Module math memfasilitasi pembuat program untuk menulis berbagai ekspresi matematis yang lebih kompleks. Misalnya untuk menghitung volume kerucut dapat digunakan rumus

$$V = \frac{1}{3}\pi r^2 t$$

Dalam ekspresi Python rumus tersebut dapat ditulis menjadi

```
V = math.pi*pow(r,2)*t*(1/3)
```

atau

```
V = (math.pi*pow(x,2)*t)/3
```

atau

```
V = (math.pow(x,2)*pi*t)/3
```

Pada penulisan ekspresi matematis, pemanggilan module math selalu diikuti dengan tanda titik dan fungsi atau konstanta yang terdapat dalam module math. Jadi untuk rumus volume kerucut tersebut setelah math. Hanya dapat diikuti dengan pi atau fungsi pow.

Latihan

1. Buat program untuk menghitung luas daerah lingkaran, dengan ukuran jari-jari merupakan inputan dari pengguna. (petunjuk: $L = \pi r^2$)

2. Buatlah program untuk menghitung nilai y jika didefinisikan $y = e^{2x} - 3x^3 + 5$, dengan nilai x inputan dari pengguna.
3. Buatlah program untuk menghitung nilai y jika didefinisikan $y = e^{-x} - \cos 2x + 2$
4. Buatlah program untuk menghitung nilai z jika didefinisikan $z = \frac{x^2+y^2}{x-y}$
5. Buatlah program untuk menghitung nilai x jika didefinisikan $x = \frac{-b+\sqrt{b^2-4ac}}{2a}$

3.5. Module fractions

Sejauh ini program yang kita diskusikan hanya melibatkan komputasi pada bilangan bulat dan bilangan desimal. Selain kedua tipe bilangan tersebut, Python menyediakan fasilitas untuk melakukan komputasi pada bilangan pecahan. Python menyediakan suatu module yang disebut module *fractions,* yang dapat menyediakan tipe bilangan baru, yaitu tipe pecahan. Tipe pecahan digunakan untuk merepresentasikan bilangan pecahan dan melakukan operasi aritmatik pada bilangan rasional, seperti

$$\frac{1}{2} + \frac{1}{3}$$

Seperti module-module yang lain, sebelum digunakan module *fractions* harus diimport ke dalam program yang dibuat, yaitu dengan perintah

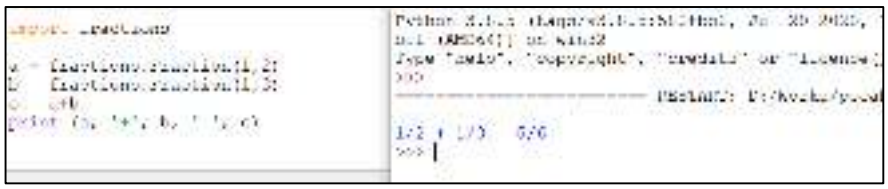
```
import fractions
```

Module *fractions* memiliki sebuah class bernama Fraction, yang berfungsi untuk memasukkan pecahan ke program Python. Fraction memerlukan dua argument, yang berperan sebagai pembilang dan penyebut dalam suatu bilangan pecahan. Contoh untuk mengisi nilai variabel a dan b masing-masing dengan nilai pecahan $\frac{1}{2}$ dan $\frac{1}{3}$, dapat dilakukan dengan penugasan

```
a = fractions.Fraction(1,2)
b = fractions.Fraction(1,3)
```

Gambar 3.19 memperlihatkan contoh penggunaan module *fractions* untuk menghitung operasi penjumlahan pada bilangan pecahan. Variabel a dan b merupakan bilangan dengan tipe fraction, yaitu $\frac{1}{2}$ dan $\frac{1}{3}$. Variabel c diisi dengan hasil penjumlahan variabel a dan b. Pada window Python

Shell di Gambar 3.19 kanan, nampak hasil eksekusi dari program pecahan.py. Dapat dilihat, karena variabel a dan b keduanya bertipe Fraction, maka hasil komputasinya yaitu variabel c juga bertipe Fraction.



Gambar 3. 19 Contoh penggunaan module fractions

Module *fractions* ini sangat bermanfaat untuk melakukan komputasi pada bilangan irasional, seperti konstanta PHI. Konstanta PHI adalah $\frac{22}{7}$ atau biasa didekati dengan 3,141592653589793…. Gambar 3.20 memperlihatkan hasil penghitungan keliling lingkaran dengan menggunakan module *fractions* dan tidak menggunakan module *fractions*.

```
import fractions
import math
r = int (input('r ='))
# Menghitung keliling lingkaran, K =2*(22/7)*r
phi = fractions.Fraction(22,7)
K = 2 * phi * r
print('phi =22/7')
print(K)
# Menghitung keliling lingkaran, K =2*(3,14...)*r
print('phi =3,14...')
K = 2 * math.pi * r
print(K)
```

Gambar 3. 20 Menghitung keliling lingkaran

Jika program pada Gambar 3.20 dieksekusi, maka akan diperoleh output seperti pada Gambar 3.21. Jika ukuran jari-jari (r) diinputkan 7, diperoleh keliling 44 untuk penghitungan menggunakan module fractions. Sedangkan jika tidak menggunakan module fractions diperoleh nilai keliling 43.982297150257104. Perbedaan hasil perhitungan ini akan bermakna dan tidaknya tergantung pada kasus yang diselesaikan. Sehingga

jika ingin hasil yang lebih akurat untuk kasus tertentu mungkin lebih baik menggunakan format bilangan pecahan.

Gambar 3. 21 Hasil eksekusi program pada Gambar 4.20

Cara lain untuk memasukkan bilangan pecahan ke program Python adalah dengan meng-import class Fraction langsung dari module *fractions* dengan statement

```
from fractions import Fraction
```

Selanjutnya untuk menggunakan class *Fraction* tidak perlu diawali dengan nama modul. Keyword *Fraction* dapat diperlakukan seperti keyword *int*, *float* ataupun *str*. Untuk lebih memahami penggunaan metode yang kedua ini, silahkan ketik ulang program pada Gambar 3.22, kemudian jalankan.

```
from fractions import Fraction
a= Fraction(input('a='))
b=Fraction(input('b='))
c=a+b
print(a, ' + ', b, ' = ', c)
```

Gambar 3. 22 Contoh penggunaan statement from import

3.1. Kesimpulan

1. Untuk menampilkan suatu output program ke layar computer, digunakan Fungsi **print()**
2. Untuk memasukkan nilai dari keyboard digunakan Fungsi **input()**
3. Module *math* merupakan module pada library Python, yang berisi konstanta dan fungsi-fungsi matematis.
4. Module *fractions,* merupakan module pada library Python yang dapat menyediakan tipe bilangan baru, yaitu tipe pecahan.

C. Latihan 3

1. Gunakan module *fractions* untuk menghitung $\frac{1}{2}^{1250}$
2. Hitung kembali soal no.1 tanpa menggunakan *fractions*, yaitu $(0.5)^{1250}$, bandingkan hasilnya.
3. Buat program dalam python untuk menghitung panjang sisi miring pada segitiga siku-siku yang panjang dua sisi siku-sikunya a dan b. (petunjuk: $c = \sqrt{a^2 + b^2}$)
4. Buat program lengkap dalam python untuk menghitung volume kerucut
5. Buat program untuk menghitung pembagian dua bilangan, tampilkan hasilnya dengan pembulatan ke atas.
6. Buat program untuk menghitung total hambatan dari rangkaian tiga hambatan listrik yang dipasang parallel. Gunakan inputan dari pengguna untuk nilai hambatan.
7. Buat program untuk mengkonversi suatu suhu dari celcius ke Fahrenheit, dengan suhu celcius merupakan inputan dari pengguna. Tampilkan output menggunakan format satuan derajat.
8. Buat program lengkap dalam Python untuk menghitung pengurangan dua bilangan pecahan sederhana, dengan kedua operan merupakan inputan dari pengguna.
9. Buat program dalam Python untuk menghitung $3a - b + 2c$, dengan a, b, dan c merupakan bilangan pecahan hasil inputan dari pengguna
10. Buat program lengkap dalam Python untuk menghitung $(e)^a - 2cosb + c$, dengan a, b, dan c bilangan pecahan inputan dari pengguna
11. Buat program yang meminta pengguna memasukkan tiga bilangan bulat. Selanjutnya program akan menampilkan hasil bagi ketiga bilangan dengan 3, dan ditampilkan dengan format 2 angka di belakang koma
12. Buat program yang meminta pengguna memasukkan satu bilangan bulat. Selanjutnya program akan menampilkan hasil bagi (//) bilangan dengan 7, dan sisa pembagian bilangan dengan 3.
13. Buat program yang meminta pengguna memasukkan lima bilangan bulat, yang akan disimpan pada variabel a, b, c, d, dan e. Program

akan menampilkan nilai yang diinputkan oleh pengguna dengan perintah **`print(a,b,c,d,e).`** Selanjutnya program akan mempertukarkan nilai dari variabel a dengan e, dan variabel b dengan d. Kemudian program akan menampilkan hasil pertukaran dengan perintah **`print(a,b,c,d,e)`** lagi.

14. Buat program dalam Python yang meminta pengguna untuk memasukkan suatu bilangan pecahan. Selanjutnya program akan menampilkan bilangan tersebut sebagai bilangan decimal, hasil bilangan tersebut setelah dibulatkan (gunakan fungsi round(bilangan), dibulatkan ke atas, dibulatkan ke bawah.
    Contoh:
    Jika dimasukkan nilai 17/8
    Maka desimalnya: 2,125
    Pembulatan : 2
    Pembulatan ke atas: 3
    Pembulatan ke bawah: 2
15. Buat program dalam Python yang meminta pengguna untuk memasukkan tiga bilangan. Selanjutnya program akan menghitung nilai rata-ratanya dan menampilkannya dalam format bilangan pecahan dan bilangan decimal 2 angka di belakang koma.

BAB 4 PERCABANGAN

A. Tujuan Pembelajaran

Tujuan yang ingin dicapai setelah pembelajaran ini adalah.

1. Diberikan suatu program yang berkaitan dengan percabangan, mahasiswa dapat menentukan output pogram tersebut secara manual dengan benar.
2. Diberikan suatu masalah, mahasiswa dapat menghasilkan program interaktif dalam Python untuk menyelesaikan masalah tersebut dengan melibatkan struktur if searah.
3. Diberikan suatu masalah, mahasiswa dapat menghasilkan program interaktif dalam Python untuk menyelesaikan masalah tersebut dengan melibatkan struktur if dua arah.
4. Diberikan suatu masalah, mahasiswa dapat menghasilkan program interaktif dalam Python untuk menyelesaikan masalah tersebut dengan melibatkan struktur if multi arah
5. Diberikan suatu masalah, mahasiswa dapat menghasilkan program interaktif dalam Python untuk menyelesaikan masalah tersebut dengan melibatkan struktur if bersarang

B. Uraian Materi

Sejauh ini, program yang ditulis mengikuti skema sederhana seperti berikut:

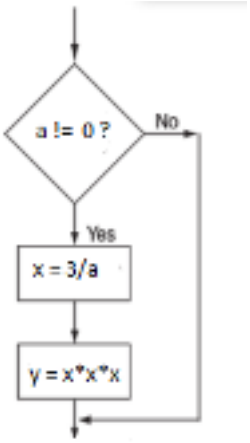
- Kumpulkan masukan dari pengguna.
- Lakukan satu atau lebih perhitungan.
- Menampilkan hasil di layar.

Pada kode program yang disusun seperti ini, pernyataan-pernyataan akan dieksekusi secara berurutan, satu demi satu, tanpa bercabang ke arah lain. Struktur program yang seperti ini disebut struktur sekuensial. Banyak algoritma yang memerlukan program untuk menjalankan beberapa pernyataan hanya dalam keadaan tertentu. Hal ini disebut dengan struktur desisi/ keputusan.

4.1. Eksekusi Bersyarat

Eksekusi bersyarat merupakan bentuk struktur keputusan yang paling sederhana, dimana tindakan atau serangkaian tindakan tertentu hanya dilakukan ketika kondisi tertentu dipenuhi. Sedangkang jika kondisinya tidak dipenuhi, maka tindakan tidak dilakukan. Bagan pada Gambar 4.1 menunjukkan gambaran logika dari eksekusi bersyarat. Simbol belah ketupat mewakili pertanyaan ya/ tidak atau kondisi benar/ salah. Jika jawaban untuk pertanyaannya adalah ya (atau jika kondisinya benar), aliran program mengikuti satu jalur yang mengarah ke tindakan yang dilakukan. Jika jawaban untuk pertanyaannya adalah tidak (atau kondisinya salah), aliran program mengikuti jalur lain yang mengabaikan tindakan.

Pada diagram alur Gambar 4.1, aktivitas menghitung "x= 3/a", " dan "y = x*x*x", hanya dilakukan ketika nilai a tidak sama dengan 0. Jika nilai a sama dengan 0, proses ini dilewati. Jadi kedua proses tersebut dilakukan secara kondisional, hanya dilakukan ketika kondisi tertentu, (x != 0), dipenuhi.



Gambar 4. 1 Contoh kasus eksekusi bersyarat

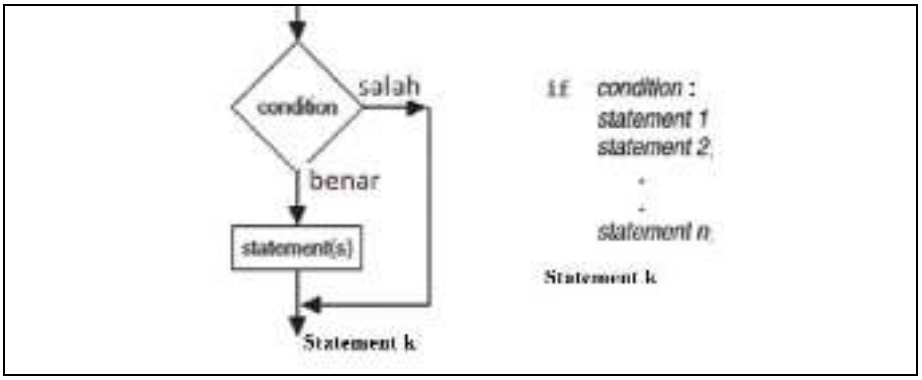
Banyak masalah kehidupan sehari-hari ataupun sains yang menggunakan struktur eksekusi bersyarat. Beberapa contoh eksekusi bersyarat dalam kehidupan sehari-hari diantaranya:

- Jika bensin mobil hampir habis, maka berhentilah di pom bensin dan dapatkan bensin.
- Jika hujan di luar, maka masuk ke dalam.
- Jika Anda lapar, maka dapatkan sesuatu untuk dimakan.

Pernyataan yang dapat digunakan untuk struktur eksekusi bersyarat adalah pernyataan **`if`**. Gambar 4.2 menunjukkan format umum pernyataan **`if`** dalam Python, sedangkan diagram alur menunjukkan secara visual cara kerja struktur **`if`** untuk eksekusi bersyarat**.**

Struktur if dimulai dengan kata kunci **`if`** sampai statement n. Perhatikan bahwa pernyataan di dalam tubuh struktur if ditulis secara indent (menjorok ke dalam). Tanda titik-dua (:) ditempatkan setelah kondisi if dan sebelum pernyataan pertama dalam tubuh if. Struktur if berakhir pada statement n. Jadi terdapat empat hal penting yang harus diperhatikan:

1. Kata if, merupakan key word Python selalu ditulis dengan furuf kecil.
2. Key word if diikuti oleh kondisi yang harus diuji, x!=0. Kondisi dapat juga ditulis dalam tanda kurung tertutup.
3. Terdapat tanda titik-dua setelah kondisi
4. Semua statement yang harus dilakukan jika kondisi dipenuhi ditulis secara indent (satu kali tab) setelah tanda titik-dua.



Gambar 4. 2 Eksekusi bersyarat

Program pada Gambar 4.3 mengilustrasikan penggunaan pernyataan if. Pengguna memasukkan satu bilangan bulat, jika bilangan yang dimasukkan bukan 0 (misalkan 5), maka program akan menampilkan pesan "bilangan Anda adalah 5", kemudian pesan "bilangan Anda bukan nol".

```
# Struktur keputusan, if1.py

x = int (input ('x='))
if x!=0:
    print('bilangan Anda adalah', x)
    print('bilangan Anda bukan nol')
print('SELESAI')
```

Gambar 4. 3 Contoh program dengan struktu reksekusi bersyarat

Yang terjadi pada program Gambar 4.3 adalah sebagai berikut: setelah pengguna memasukkan nilai untuk x, misalnya dengan mengetik angka 5, selanjutnya program akan mengevaluasi nilai dari ekpresi relasional x!=0. Karena x bernilai 5, maka ekpresi x!=0 bernilai benar. Akibatnya program akan melaksanakan penugasan untuk memunculkan ke layar string "bilangan Anda adalah 5" dan pada baris berikutnya memunculkan string "bilangan Anda bukan nol". Selanjutnya program ke luar dari struktur if dan melaksanakan penugasan berikutnya yaitu menampilkan pesan "SELESAI". Jika pengguna memasukkan nilai 0 untuk nilai x, maka ekspresi x!=0 bernilai salah, akibatnya program tidak akan masuk ke dalam struktur if melainkan langsung melaksanakan penugasan untuk menampilkan pesan "SELESAI".

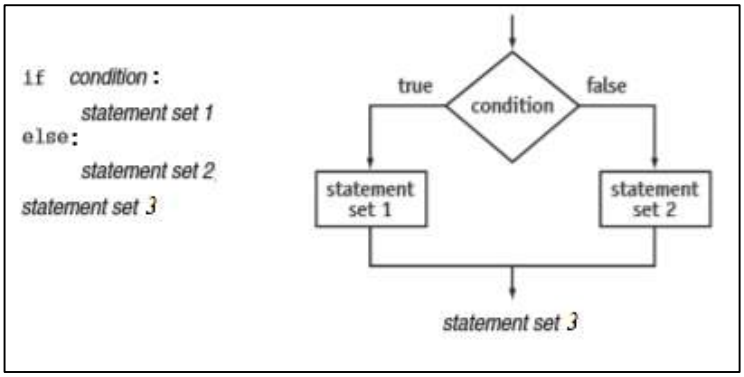
Struktur if berakhir ketika suatu statement ditulis tanpa indentasi. Jadi pada program Gambar 4.3, struktur if berakhir pada perintah print yang kedua. Sehingga perintah print yang terakhir akan selalu dieksekusi berapapun nilai x yang dimasukkan. Gambar 4.4 memperlihatkan contoh menghitung nilai z jika didefinisikan $z=\frac{x^2+y^2}{x-y}$. Karena nilai penyebut tidak boleh nol, maka nilai z dapat dihitung hanya jika nilai x-y tidak nol.

```
x = eval (input('x ='))
y = eval (input('y ='))
if (x-y!=0):
    z = (x**2+y**2)/(x-y)
    print(z)
print ('SELESAI')
```

Gambar 4. 4 Contoh lain penggunaan struktur eksekusi bersyarat

4.2. Eksekusi Alternatif

Eksekusi alternatif atau keputusan dua arah merupakan pengembangan dari struktur eksekusi bersyarat. Untuk mengimplementasikan struktur eksekusi alternatif pada program Python, dapat digunakan struktur if-else. Format struktur if-else dapat dilihat pada Gambar 4.5.



Gambar 4. 5 Format struktur eksekusi alternatif

Saat masuk struktur if, nilai dari kondisi akan dievaluasi. Selanjutnya jika kondisi bernilai benar, maka statement set 1 pada Gambar 4.5 akan dilaksanakan. Jika kondisi bernilai salah, maka yang akan dilaksanakan adalah statement set 2. Perhatikan bahwa statement dalam header if dan statement dalam header else ditulis indent. Statement set 3 ditulis tidak indent, artinya statement tersebut berada di luar header else.

Salah satu contoh penerapan struktur eksekusi alternatif adalah algoritma menentukan apakah suatu bilangan ternasuk bilangan genap atau bilangan ganjil. Suatu bilangan bulat hanya memiliki satu status, yaitu

merupakan bilangan genap atau bilangan ganjil, tidak akan kedua-duanya. Suatu bilangan bulat merupakan bilangan genap, jika sisa pembagian bilangan tersebut dengan 2 nilainya nol, jika tidak maka merupakan bilangan ganjil. Gambar 4.6 memperlihatkan contoh program penggunaan struktur eksekusi alternative, untuk menentukan apakah suatu bilangan merupakan bilangan genap atau bilangan ganjil.

```
bil = eval (input('Bilangan ='))
SisaBagi = bil % 2
if (SisaBagi==0):
    print(bil, ' merupakan bilangan genap')
else:
    print(bil, ' merupakan bilangan ganjil')
print('SELESAI')
```

Gambar 4. 6 Contoh penerapan struktur eksekusi alternatif if-else

Saat program pada Gambar 4.6 dieksekusi, kemudian pengguna memasukkan 7 untuk nilai variabel bil. Program akan menghitung sisa pembagian 7 dengan 2 (diperoleh 1), kemudian menyimpan hasilnya pada variabel SisaBagi. Selanjutnya kondisi dievaluasi, ekspresi SisaBagi==0 bernilai salah (karena SisaBagi bernilai 1), shingga akan dilaksanakan penugasan untuk memunculkan string "7 bilangan ganjil" ke layar.

Jika ada lebih dari satu kondisi yang harus dievaluasi, dapat digunakan operator logika ***and*** atau ***or***. Misalkan untuk menentukan seorang siswa lulus dengan predikat cum laude harus dipenuhi dua syarat, yaitu memiliki IPK ≥3,51 dan masa studinya maksimal 4 tahun. Dengan inputan nilai IPK dan masa studi dapat ditulis

```
if (IPK >=3.51 and masa_studi <=4.0):
    print('Lulus dengan predikat Cum Laude')
else:
    print('Lulus tanpa predikat Cum Laude')
```

Pada kasus ini karena semua syarat harus dipenuhi, maka digunakan operator logika *and*. Jika tidak harus kedua/semuanya dipenuhi dapat digunakan operator logika *or*.

Suatu *header* dalam Python adalah suatu keyword tertentu yang diikuti dengan tanda titik dua. Pada contoh Gambar 4.6 terdapat dua *header,* yang terdiri dari.

1. if (SisaBagi==0): (memiliki keyword if)
2. else: (memiliki keyword else)

Himpunan statement yang mengikuti suatu *header* disebut *suite/block*. Semua statement pada suite yang sama harus ditulis inden (menjorok) dengan ukuran yang sama. Setiap pasangan *header* dengan *suite* yang bersesuaian disebut *clause*. Rangkaian *clause* yang bersesuaian disebut *compound statement*. Jadi satu *compound statement*, memiliki lebih dari satu *clause*.

Latihan

1. Temukan kesalahan pada potongan program berikut dan jelaskan apa kesalahannya.

```
a. waktu = 12
   if waktu > 40;
      print (" Waktu anda sudah habis")
b. Nilai = 50
   if Nilai = 70:
      print("Anda mencapai batas minimum lulus.")
c. total = 500
   if (total > 400):
      print("Anda mendapat bonus 50.\n")
   total = total + 50
```

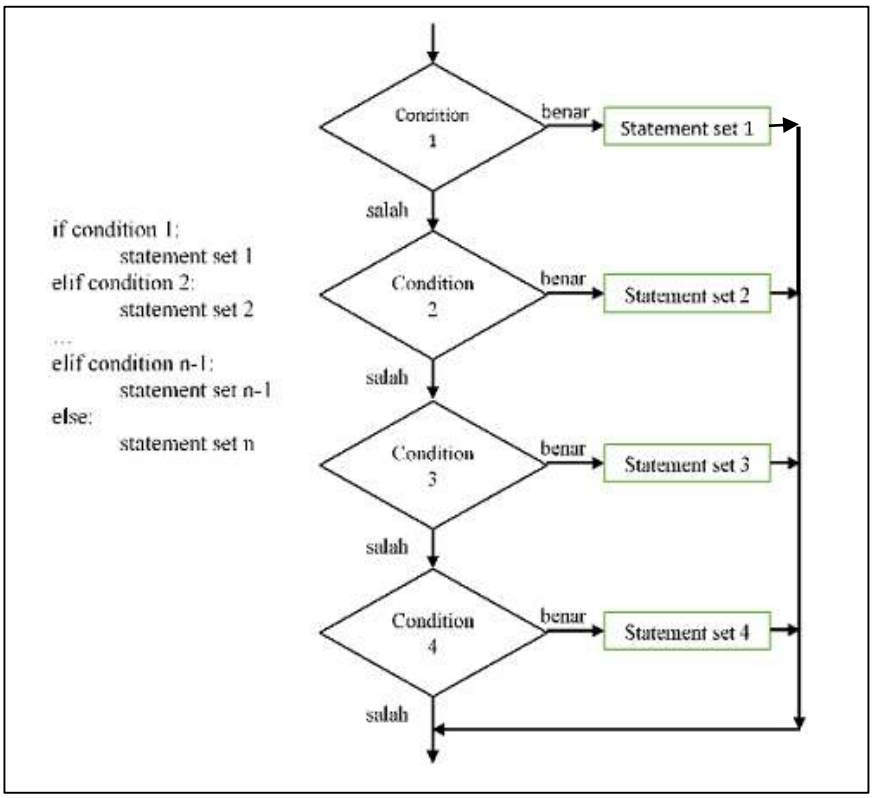
2. Tulis pernyataan if yang menyimpan nilai 0 ke x jika y sama dengan 10.
3. Tulis pernyataan if/else yang menyimpan nilai 1 ke x jika y sama dengan 100. Jika tidak simpan nilai 0 ke x.

4.3. Syarat Berantai

Syarat berantai merupakan struktur percabangan yang memberikan alternatif arah lebih dari dua. Terdapat berbagai masalah yang memerlukan struktur syarat berantai. Seperti saat kita akan menentukan apakan suatu bilangan merupakan bilangan positif, bilangan negative atau nol. Dalam hal ini terdapat tiga alternative pilihan arah program. Untuk kasus-kasu

seperti ini dalam Python dapat digunakan struktur if dengan banyak arah seperti pada Gambar 4.7.

Pada struktur syarat berantai seperti pada Gambar 4.7, ketika memasuki struktur if, program akan mengevaluasi condition-1, jika nilainya benar maka akan dikerjakan penugasan pada statement set-1. Jika tidak benar, maka lanjutkan evaluasi condition-2. Jika condition-2 nilainya benar, maka akan dikerjakan penugasan pada statement set-2, jika tidak benar lanjutkan evaluasi condition-3. Demikian seterusnya sampai akhirnya evaluasi condition n-1, jika benar kerjakan penugasan pada statement set n-1, jika tidak kerjakan statement set n.



Gambar 4. 7 Format struktur syarat berantai

Walaupun banyak pilihan arah, tetapi dalam struktur berantai hanya satu stement set yang akan dikerjakan, yaitu yang sesuai dengan kondisinya. Perhatikan bahwa struktur syarat berantai selalu diawali dengan keyword **`if`**, diikuti dengan satu atau beberapa keyword **`elif`**, dan sebagai penutup dari struktur ini selalu diakhiri dengan keyword **`else`**. Kasus berikut merupakan contoh penerapan struktur syarat berantai. Kualifikasi perolehan skor akan dikelompokkan ke dalam lima tingkatan, yaitu

skor ≥ 80, kriteria sangat baik
80 > skor ≥ 70, kriteria baik
70 > skor ≥ 60, kriteria cukup
60 > skor ≥ 50, kriteria kurang
50 > skor, kriteria sangat kurang

Pada kasus ini, kondisi yang pertama dievaluasi adalah ekspresi relasional skor ≥ 80, jika ekspresi ini bernilai salah selanjutnya ekspresi yang kedua dievaluasi, yaitu skor ≥70. Jika ekspresi ini bernilai salah, dilanjutkan dengan evaluasi untuk ekspresi ketiga, yaitu skor ≥ 60, dan seterusnya. Gambar 4.8 memperlihatkan alternatif program lengkap untuk kasus ini.

```
skor = int(input("skor = "))
if skor >= 80:
    print("Skor = ", skor, "\nKriteria: Sangat Baik")
elif skor >= 70:
    print("Skor = ", skor, "\nKriteria: Baik")
elif skor >= 60:
    print("Skor = ", skor, "\nKriteria: Cukup")
elif skor >= 50:
    print("Skor = ", skor, "\nKriteria: Kurang")
else:
    print("Skor = ", skor, "\nKriteria: Sangat Kurang")
```

Gambar 4. 8 Contoh penggunaan struktur percabangan berantai

Masing-masing kondisi dalam struktur, tergantung pada nilai semua kondisi sebelumnya. Pernyataan setelah elif tertentu, diekseskusi ketika ekspresi relasional yang mengikuti elif bernilai benar dan semua ekspresi relasional sebelumnya bernilai salah. Misalnya jika pengguna

memasukkan nilai skor 60, maka ekspresi relasional `skor >= 80,` dan ekspresi `skor >= 70` bernilai salah. Sedangkan ekrepresi relasional `skor >= 60` bernilai benar. Sehingga penugasan didalamnya, yaitu `print("Skor = ", skor, "\nKriteria: Cukup")` akan dieksekusi.

Kenapa harus menggunakan struktur percabangan perantai? Bagaimana kalau menggunakan if berulang? Untuk memahaminya silahkan anda bandingkan dengan program di bawah ini. Apakah keduanya dapat dinyatakan dengan struktur if berulang?

```
bilangan=int(input())
if (bilangan > 0:
      print('Positif')
elif (bilangan < 0):
      print('Negatif')
else:
      print('Nol')
```

Gambar 4. 9 Penggunaan struktur percabangan berantai lainnya

4.4. Kondisi Bersarang

Suatu struktur percabangan mungkin berada di dalam struktur percabangan yang lain. Struktur yang demikian disebut kondisi bersarang, atau if bersarang. Di dalam struktur eksekusi alternative misalnya, mungkin akan ada struktur eksekusi alternative yang lain, contoh.

```
if x==y:
    print('x sama dengan y')
else:
    if x<0:
        print('x kurang dari y')
    else:
        print('x lebih y')
```

Kondisi yang terluar memiliki dua cabang. Cabang pertama berisi satu pernyataan sederhana. Cabang yang kedua berisi struktur if lain yang memiliki dua cabang, dan masing-masing cabang berisi satu pernyataan sederhana. Struktur kondisi bersarang dapat diterapkan pada program untuk menentukan akar persamaan kuadrat, pemilihan menu dan lain-lain.

Untuk kasus tertentu operator logika dapat menyederhanakan struktuk percabangan yang digunakan. Struktur

```
if  0 < x:
    if x <10:
        print('x bilangan positif satu digit')
```

dapat disederhanakan dengan memanfaatkan operator logika and, sebagai berikut

```
if  0 < x and x <10:
    print('x bilangan positif satu digit')
```

Untuk kondisi ini, python menyediakan cara yang lebih ringkas, yaitu

```
if  0 < x <10:
    print('x bilangan positif satu digit')
```

Tentu saja tidak semua struktur percabangan bersarang dapat disederhanakan seperti kasus di atas. Salah satu contoh kasus yang dapat diimplementasikan dengan struktur percabangan bersarang adalah menentukan akar persamaan kuadrat. Bentuk umum persamaan kuadrat adalah

$ax^2 + bx + c = 0$ dengan $a \neq 0$

Sehingga akar-akar dari persamaan kuadrat dapat dicari jika $a \neq 0$. Setelah itu akar persamaan kuadrat dapat dicari menggunakan rumus yaitu

$x_1 = \frac{-b+\sqrt{D}}{2a}$; $x_2 = \frac{-b-\sqrt{D}}{2a}$ dengan syarat $D = b^2 - 4ac \geq 0$.

Sehingga alternative algoritma yang dapat disusun adalah dengan struktur percabangan bersarang.

```
Input: a, b, c
If a!=0
     D=b*b-4*a*c
     If D > 0
        X1=(-b+sqrt(D))/(2*a)
        X2=(-b-sqrt(D))/(2*a)
     Else if D==0
        X1= -b/(2*a)
        X2=X1
     Else
        Akar imajiner
Else
     Bukan persamaan kuadrat
```

4.5. Penulisan Percabangan dalam Satu Baris

Penulisan percabangan juga dapat ditulis dalam satu baris. Pernyataan if pada contoh berikut

```
if (a < B):
    print("Salaam")
```

dapat ditulis dalam satu baris menjadi

```
print("Salaam") if (a < B)
```

Contoh lainnya

```
if (a < B):
    print("Salaam")
else:
    print("Hello")
```

menjadi

```
print("Salaam") if (a < B) else print("Hello")
```

Masing-masing cara penulisan memiliki kelebihan dan kekurangannya. Silahkan digunakan mana yang menurut anda paling sesuai untuk program anda.

4.6. Kesimpulan

Hal yang dapat disimpulkan dari pembahasan ini:

1. Eksekusi bersyarat merupakan bentuk struktur keputusan yang paling sederhana, dimana tindakan atau serangkaian tindakan tertentu hanya dilakukan ketika kondisi tertentu dipenuhi. Sedangkang jika kondisinya tidak dipenuhi, maka tindakan tidak dilakukan, dengan format umum

   ```
   if kondisi:
       Kumpulan Pernyataan
   ```

2. Eksekusi alternatif atau keputusan dua arah, serangkaian tindakan tertentu hanya dilakukan ketika kondisi tertentu dipenuhi. Sedangkang jika kondisinya tidak dipenuhi, maka dilakukan serangkaian tindakan yang lain, dengan format umum

   ```
   if kondisi:
       Kumpulan Pernyataan-1
   else:
       Kumpulan Pernyataan-2
   ```

3. Syarat berantai merupakan struktur percabangan yang memberikan alternatif arah lebih dari dua, dengan format

```
if kondisi-1:
       Kumpulan Pernyataan-1
elif kondisi-2:
       Kumpulan Pernyataan-2
elif kondisi-3:
       Kumpulan Pernyataan-3
….
else:
       Kumpulan Pernyataan-n
```

4. Kondisi bersarang merupakan struktur percabangan di dalam struktur percabangan yang lain, contoh format

```
if kondisi-1:
       if kondisi:
         Kumpulan Pernyataan
       else:
         Kumpulan Pernyataan
elif kondisi-2:
       if kondisi:
         Kumpulan Pernyataan
else:
       kumpulan pernyataan
```

C. Latihan 4

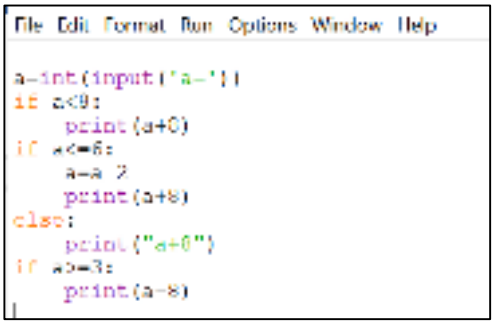
1. Tulis pernyataan if yang akan mengalikan total dengan 1,5 jika waktu lebih dari sama dengan 30.
2. Tulis pernyataan if yang menyimpan nilai 0,20 ke variabel bonus jika penjualan lebih dari 5 juta.
3. Tulis pernyataan if yang akan menampilkan kalimat "Anda Lulus", jika nilai lebih dari sama dengan 12, dan kurang dari 16.
4. Tulis pernyataan if yang akan menyimpan nilai akar dari variabel x ke variabel y, jika nilai x tidak negative.
5. Tulis pernyataan if/else yang akan menyimpan nilai 0,10 ke bonus jika penjualan kurang dari sama dengan 50000, jika tidak bonus 0,20.
6. Tulis pernyataan if/else untuk menampilkan pesan "Anda lulus", jika nilai variabel A atau variabel B lebih dari 80, jika tidak maka menampilkan pesan "Anda tidak lulus".

7. Apakah pernyataan if/else di bawah ini akan berfungsi sama dengan dua pernyataan if yang terpisah? Jelaskan!

```
if x < y:                if x < y:
    print(‘ 1’)              print(‘ 1’)
if x > y:                else:
    print(‘ 2’)              print(‘2)
```

8. Buat program lengkap dalam Python. Program akan menampilkan satu kata dalam bahasa Indonesia, dan menanyakan kepada pengguna terjemahan kata tersebut dalam Bahasa Inggris. Jika jawaban benar maka akan muncul pesan ucapan selebrasi, jika tidak akan muncul pesan penyemangat.
9. Buat program lengkap dengan spesifikasi:
   Pengguna memasukkan nilai tugas, UTS, dan UAS. Selanjutnya akan dihitung nilai rata-ratanya. Jika nilai rata-rata lebih dari sama dengan 75, maka akan ditampilkan nilai rata-ratanya dan pesan “Selamat, Anda lulus”
   Jika tidak, maka akan ditampilkan nilai rata-ratanya dan pesan “Anda harus belajar lebih giat”
10. Buat program yang akan menampilkan pesan “Sangat Baik”, jika nilai A dan B keduanya lebih dari sama dengan 80, jika tidak tampilkan pesan “Belum Berhasil”.
11. Buat program untuk menjumlahkan dua bilangan pecahan. Kedua bilangan pecahan merupakan inputan dari pengguna. Jika pengguna memasukkan bilangan decimal, program akan mengeluarkan pesan kesalahan.
12. Buat program dalam Bahasa Python dengan spesifikasi: Program akan meminta pengguna memasukkan ukuran balok. Selanjutnya program akan menanyakan ke pengguna berapa volum balok tersebut. Program akan memeriksa apakah jawaban pengguna benar atau tidak. Jika jawabannya benar pengguna mendapat nilai 100, jika tidak pengguna mendapat nilai 10.

13. Perhatikan program di bawah ini.

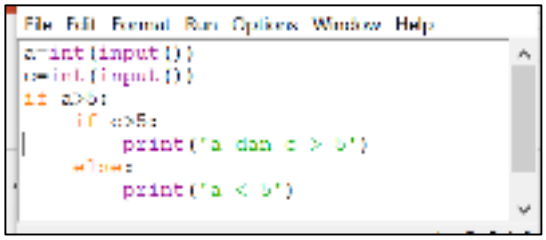


Gambar 4. 10 Program Latihan 4.13

Gunakan Program pada Gambar 4.10 untuk menjawab secara manual pertanyaan-pertanyaan di bawah ini

a. Tentukan hasil dari program, jika variabel a bernilai 4
b. Tentukan hasil dari program, jika variabel a bernilai 9
c. Tentukan hasil dari program, jika variabel a bernilai 7
d. Tentukan hasil dari program, jika variabel a bernilai 5

14. Jelaskan kesalahan yang ada pada program Gambar 4.11, berdasarkan hasil analisis anda bagaimana perbaikan yang mungkin dilakukan pada program tersebut.



Gambar 4. 11 Program Latihan 4.14

15. Diketahui aturan penggajian yang digunakan oleh suatu perusahaan adalah sebagai berikut:

a. untuk 40 jam pertama, gaji dihitung dengan rumus golongan gaji dikali jam kerja

b. untuk kelebihan jam kerja setelah 40 jam pertama bekerja, perusahaan membayar satu setengah kali golongan gaji.

Buatlah program untuk membantu perusahaan dalam menghitung gaji pegawai per minggu.

16. Suatu toko menerapkan aturan, jika seorang pembeli memiliki kartu anggota dan memiliki nilai poinnya lebih dari 500 maka mendapat diskon 5% dari total yang harus dibayar. Jika pembeli memiliki kartu anggota tetapi poin yang dikumpulkannya di bawah 500 maka dia mendapat diskon 2% dari total yang harus dibayar. Jika bukan anggota tidak mendapat diskon. Buatlah program dalam Python yang dapat digunakan untuk menghitung total pembayaran setiap pembeli.
17. Buat program "mesin genap-ganjil". Program akan meminta pengguna memasukkan satu bilangan bulat. Jika bilangan yang diinputkan bilangan genap, maka akan muncul pesan 'bilangan genap', kemudian akan diikuti dengan 5 bilangan genap berikutnya. Jika yang diinputkan bilangan ganjil, maka akan muncul pesan 'bilangan ganjil', kemudian diikuti dengan 5 bilangan ganjil berikutnya. Jika inputan bukan bilangan bulat akan muncul pesan kesalahan.
18. Buat program untuk konversi suhu antara Celcius, Fahrenheit, Kelvin, Reamur. Pengguna dapat memilih inputan suhu, dan akan dikonversi ke mana.
19. Buat program dalam Python untuk menentukan apakah suatu tahun merupakan tahun kabisat atau bukan. Aturan untuk menentukan apakah suatu tahun termasuk tahun kabisat atau bukan sebagai berikut:
    a. Jika angka tahun itu habis dibagi 400, maka tahun itu sudah pasti tahun kabisat.
    b. Jika angka tahun itu tidak habis dibagi 400 tetapi habis dibagi 100, maka tahun itu sudah pasti bukan merupakan tahun kabisat.
    c. Jika angka tahun itu tidak habis dibagi 400, tidak habis dibagi 100 akan tetapi habis dibagi 4, maka tahun itu merupakan tahun kabisat.

d. Jika angka tahun tidak habis dibagi 400, tidak habis dibagi 100, dan tidak habis dibagi 4, maka tahun tersebut bukan merupakan tahun kabisat.

20. Gunakan struktur if untuk membuat proram yang akan menampilkan kata dalam Bahasa Indonesia. Selanjutnya program akan menanyakan arti kata tersebut dalam Bahasa Inggris kepada pengguna, dan pengguna akan mengetikkan jawabannya. Jika jawaban salah, program akan menampilkan pertanyaan berikutnya. Jika jawabannya benar, program selesai. Banyaknya pertanyaan minimal 4.

Bab 5 FUNCTION

A. Tujuan Pembelajaran

Tujuan yang ingin dicapai setelah pembelajaran ini adalah.

1. Diberikan suatu program yang memuat fungsi, mahasiswa dapat menjelaskan cara kerja program tersebut dengan benar.
2. Diberikan suatu masalah, mahasiswa dapat menghasilkan program untuk menyelesaikan masalah tersebut dengan melibatkan penggunaan fungsi dalam Python dengan kreatif, dan dapat mempresentasikannya dengan bertanggung jawab.
3. Diberikan suatu fungsi yang didefinisikan secara rekursif, mahasiswa dapat menghasilkan program dalam Python untuk mengimplementasikan fungsi tersebut dengan benar.

B. Uraian Materi

Sejauh ini beberapa feature Python yang paling sering digunakan adalah variabel, ekspresi, percabangan, pengulangan, input dan print. Secara teori hanya diperlukan beberapa instruksi untuk menuliskan program. Akan tetapi instruksi saja tidak cukup. Ketika masalahnya semakin kompleks, mungkin suatu program tidak diimplementasikan oleh satu orang, diperlukan tim programmer untuk membangun suatu projek yang besar.

Dalam kehidupan sehari-hari, untuk memudahkan dalam pengaturan, suatu projek besar biasanya dipecah menjadi projek-projek kecil. Masing-masing subprojek akan fokus menyelesaikan submasalah secara terpisah. Demikian pula dalam pemrograman. Program yang besar dapat dibagi menjadi program-program kecil yang disebut *program routines* (atau *routines*). Routines merupakan bagian yang penting dalam programing.

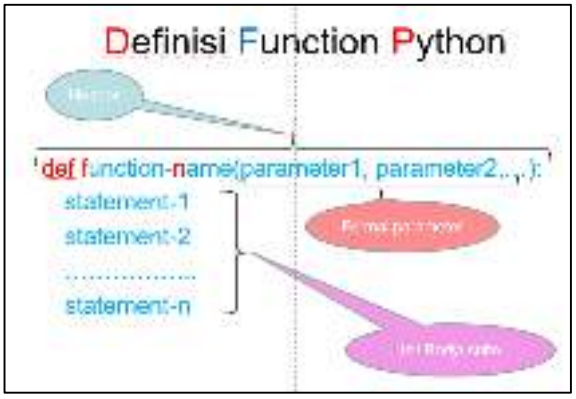
Program routines dalam Python disebut fungsi (*functions)*. Suatu routines merupakan nama dari kumpulan statements untuk melaksanakan tugas tertentu. Fungsi dibuat untuk menjadikan program lebih terstruktur sehingga mudah dipahami dan mudah dikembangkan.

5.1. Menambahkan Fungsi Baru

Bagian ini kita akan belajar bagaimana membuat fungsi milik kita yang dapat digunakan seperti fungsi pada library. Fungsi biasanya digunakan untuk memecah masalah menjadi potongan-potongan kecil yang bisa dikelola, atau disebut juga modul. Daripada menulis satu fungsi yang panjang yang memuat semua statements yang diperlukan untuk menyelesaikan masalah, akan lebih baik jika ditulis beberapa fungsi yang lebih kecil, dengan masing-masing untuk menyelesaikan bagian tertentu yang spesifik dari masalah. Fungsi-fungsi yang kecil ini dapat dieksekusi dalam keseluruhan aplikasi untuk menyelesaikan masalah. Pendekatan ini disebut *divide and conquer* karena membagi masalah yag besar menjadi beberapa masalah yang lebih kecil yang lebih mudah diselesaikan. Selain itu, penulisan fungsi dapat menyederhanakan program. Jika tugas tertentu dilakukan di beberapa tempat dalam suatu program, suatu fungsi dapat ditulis satu kali untuk melakukan tugas itu, dan kemudian dieksekusi kapan saja diperlukan.

Definisi fungsi memuat pernyataan-pernyataan yang membangun fungsi tersebut. Gambar 5.1 memperlihatkan struktur dari definisi fungsi pada Python. Ketika membuat suatu fungsi, kita harus menuliskan definisinya. Semua definisi fungsi memiliki bagian-bagian:

1. Header fungsi diawali keyword def, diikuti dengan nama fungsi, dan daftar parameter di dalam tanda kurung, diakhiri dengan tanda titik dua.
    a. Keyword def: menunjukkan bahwa akan didefinisikan suatu fungsi.
    b. Aturan penamaan fungsi sama dengan aturan penamaan variabel.
    c. Daftar parameter adalah daftar variabel yang menyimpan nilai yang diteruskan ke fungsi. Jika tidak ada nilai yang diteruskan ke fungsi, daftar parameternya kosong. Banyaknya parameter menunjukkan banyaknya argument yang diperlukan saat pemanggilan fungsi.
2. Body fungsi, merupakan seperangkat pernyataan yang berguna untuk menjalankan tugas yang diberikan pada fungsi tersebut. Pernyataan-pernyataan ini ditulis dengan indentasi.



Gambar 5. 1 Struktur Fuction

Suatu fungsi dapat memiliki balikan suatu nilai atau balikan lain (*side effect*). Balikan dari fungsi pow(x,y) adalah nilai perpangkatan x dengan y. Jika suatu fungsi memiliki balikan *side effect*, maka fungsi tersebut hanya bertugas menampilkan suatu nilai atau pesan ke layar, seperti fungsi print().

Suatu fungsi dapat memiliki parameter ataupun tidak. Berikut adalah contoh definisi fungsi sederhana. Nama fungsinya Salam. Fungsi ini tidak memiliki parameter, dan tugasnya adalah menampilkan pesan sapaan kepada pengguna di layar computer.

```
def Salam():
  print('Selamat Datang')
  print('Anda berada di kelas Pemrograman Dasar 1')
```

Berikut ini adalah contoh fungsi yang memiliki dua parameter, dengan nama fungsi jumlahKuadrat. Fungsi ini menghasilkan suatu nilai, yaitu jumlah dari kuadrat argumen-argumennya.

```
def jumlahKuadrat (a,b):
        x = a**2+b**2
        return x
```

5.2. Nilai Balikan

Jika suatu fungsi memiliki balikan suatu nilai, maka fungsi tersebut dapat menghasilkan suatu nilai yang disebut nilai balikan atau *return value*. Pernyataan return dalam fungsi merupakan suatu ekspresi. Pernyataan ini

mengandung makna “Kembali segera dari fungsi ini dan gunakan ekspresi berikut sebagai nilai pengembalian”. Kembali segera dari fungsi, maksudnya kembali ke fungsi yang memanggil. Pernyataan ini menyebabkan fungsi berakhir, dan mengirimkan nilai variabel hasil kembali ke pernyataan yang memanggil fungsi. Fungsi pengembalian nilai harus memiliki pernyataan pengembalian yang ditulis dalam format umum berikut:

```
return ekspresi
```

Dalam format umum, ekspresi adalah nilai yang akan dikembalikan. Itu bisa berupa ekspresi apa pun yang memiliki nilai, dapat berupa nilai, variabel, atau ekspresi matematika. Nilai ekspresi dikembalikan oleh fungsi, dan dikirim kembali ke pernyataan yang memanggil fungsi. Berikut contoh cara penulisan pengembalian fungsi dengan suatu nilai dan ekspresi matematika.

```
def pembagian (bil1, bil2):
        if (bil2==0):
            return 0
        else:
            return bil1/bil2+3
```

Pada definisi fungsi di atas, fungsi pembagian memiliki dua parameter bil1 dan bil2. Jika nilai bil2 nol, maka fungsi pembagian akan mengembalikan nilai 0. Jika tidak, maka fungsi pembagian akan mengembalikan nilai dari ekspresi **`bil1/bil2+3.`**

5.3. Pemanggilan Fungsi

Pemanggilan fungsi adalah suatu pernyataan yang menyebabkan fungsi dieksekusi. Jadi suatu fungsi hanya dieksekusi ketika dipanggil. Sejauh ini kita telah menggunakan beberapa fungsi pada library Python seperti **`print(), input(),int()`** dan **`pow()`**. Kita biasa menggunakan fungsi dengan cara memanggil fungsi tersebut, seperti berikut.

```
>>> pow(2,3)
8
```

Nama fungsi adalah **pow**. Ekspresi yang ditulis di dalam tanda kurung disebut argument dari fungsi. Hasil/ balikan dari fungsi ini adalah nilai bilangan bulat 8. Jadi suatu fungsi memerlukan argument dan akan

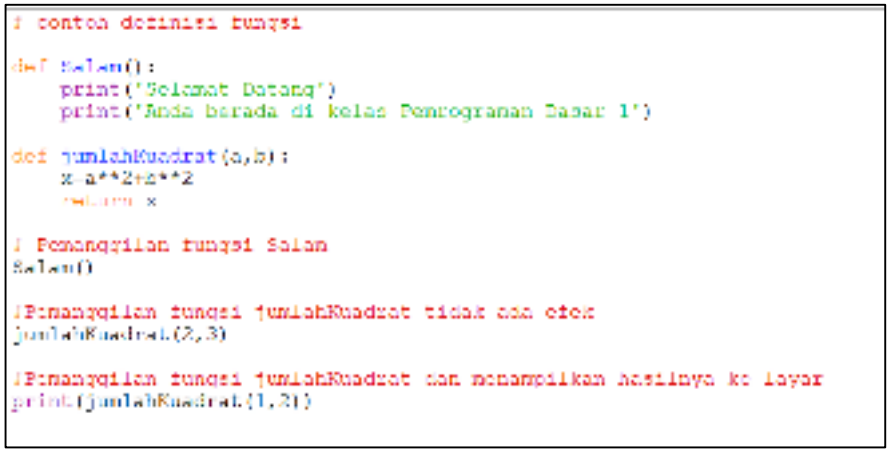
memberikan hasil/ balikan. Hasil atau balikan dari suatu fungsi disebut nilai balikan (*return value*). Pada contoh ini fungsi pow memerlukan dua argument berupa bilangan. Jika argument yang diberikan tidak sesuai, maka akan muncul pesan kesalahan, seperti contoh berikut.

```
>>> pow(2)
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    pow(2)
TypeError: pow() missing required argument 'exp' (pos 2)
```

Pemanggilan fungsi juga dapat dilakukan dengan melibatkan nilai dari variabel, ataupun dengan ekspresi lain, berikut adalah beberapa contoh pemanggilan fungsi.

```
>>> a=2;b=4
>>> pow(a,b)
16
>>> pow(a-1,b+1)
1
>>> pow(int(input()),3)
3
27
```

Cara pemanggilan fungsi pada mode script berbeda dengan pemanggilan fungsi pada Shell Python, khususnya untuk fungsi yang memiliki balikan suatu nilai. Gambar 5.2 memperlihatkan suatu program yang memiliki dua fungsi, yaitu fungsi Salam dan fungsi jumlahKuadrat.



Gambar 5. 2 Contoh definisi dan pemanggilan fungsi pada mode script

Ketika suatu fungsi dipanggil, program bercabang ke fungsi yang dipanggil. Selanjutnya program akan menjalankan semua penugasan yang ada di dalam fungsi tersebut. Pada Gambar 5.2, pernyataan **`Salam(),`** merupakan pemanggilan fungsi Salam(). Ketika dieksekusi program akan masuk ke dalam fungsi Salam dan mengerjakan penugasan di dalamnya, yaitu menampilkan pesan ke layar. Setelah selesai mengerjakan semua tugas di dalam fungsi Salam, program akan kembali ke fungsi utama dan melanjutkan eksekusi ke baris berikutnya, yaitu pemanggilan fungsi jumlahKuadrat dengan argument 2 dan 3. Selanjutnya program akan masuk ke fungsi jumlahKuadrat dan melaksanakan semua tugas yang ada di dalam fungsi tersebut, yaitu mengisi nilai variabel x, kemudian mengirimkannya sebagai nilai balikan. Setelah selesai program akan kembali ke fungsi utama dan melanjutkan eksekusi ke baris berikutnya, yaitu menampilkan hasil pemanggilan fungsi jumlahKuadrat dengan argument 1 dan 2. Selanjutnya program akan masuk kembali ke fungsi jumlahKuadrat dan melaksanakan semua tugas yang ada di dalam fungsi tersebut, yaitu mengisi nilai variabel x, kemudian mengirimkannya sebagai nilai balikan. Setelah selesai kembali ke fungsi utama akan menampilkan hasil balikan dari fungsi jumlahKuadrat, yaitu 5.

Karena nilai balikan dari fungsi adalah suatu nilai, maka nilainya dapat langsung ditampilkan di layar seperti pada Gambar 5.2 atau

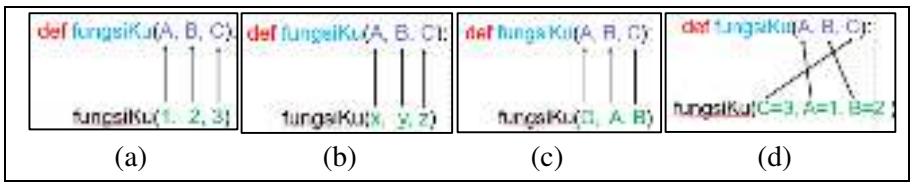
digunakan dalam komputasi berikutnya. Gambar 5.3 memperlihatkan contoh penggunaan nilai balikan dari fungsi untuk komputasi pada fungsi pemanggil.

```
'Isi variabel c dengan nilai balikan dari fungsi pembagian dengan argumen a dan b
c=pembagian(a,b)
'''
Isi variabel d dengan nilai balikan dari fungsi pembagian dengan argumen a dan c
d variabel dengan nilai balikan dari fungsi pembagian dengan argumen b dan c
'''
d=pembagian(a,c)/pembagian(b,c)
print(a,b,c,d)
```

Gambar 5. 3 Contoh penggunaan nilai balikan fungsi dalam komputasi

5.4. Parameter dan Argument

Parameter adalah variabel khusus yang menyimpan nilai yang dikirimkan ke dalam fungsi. Nilai yang dikirim ke suatu fungsi ini disebut dengan argumen. Nilai yang dikirim sebagai argumen di dalam tanda kurung pemanggilan fungsi akan disalin ke dalam variabel parameter pada fungsi yang dipanggil. Gambar 5.4 memberikan ilustrasi hubungan parameter pada header fungsi dengan argument pada pemanggilan fungsi.



Gambar 5. 4 Hubungan Parameter dan argumen fungsi

Pada Gambar 5.4, cara pemanggilan fungsi pada gambar (a), (b), dan (c) akan menyebabkan setiap argumen yang dikirimkan kepada fungsiKu akan mengisi nilai dari variabel parameter pada fungsiKu sesuai dengan urutannya. Dengan pemanggilan cara ini, kita harus tahu urutan argument yang benar saat pemanggilan fungsi. Contohnya jika kita akan menghitung nilai dari 3^4 menggunakan fungsi pow, maka kita harus memanggil fungsi pow dengan argument pertama 3, dan argument kedua 4, yaiut pow(3,4). Jika argumen yang dikirimkan urutannya terbalik, maka yang dihitung

oleh fungsi pow adalah nilai dari 4^3, sehingga hasilnya tidak sesuai dengan yang diharapkan.

Cara pemanggilan fungsi seperti pada Gambar 5.4 (d) adalah pemanggilan fungsi dengan argument penugasan. Hal ini diperbolehkan dalam Python. Dengan cara ini pengiriman argument tidak didasarkan pada urutan, tetapi berdasarkan nama variabelnya. Pernyataan fungsiKu(C=3, A=1, B=2) bermakna isikan nilai 3 ke variabel C, nilai 1 ke variabel A, nilai 2 ke variabel B yang ada pada fungsiKu. Jadi dalam hal ini variabel A, B, dan C merupakan variabel milik fungsiKu bukan milik fungsi pemanggilan. Pernyataan pemanggilan fungsi dengan argument suatu penugasan dapat juga menggunakan inputan dari pengguna, seperti contoh berikut. Apakah dua pemanggilan fungsi ini merupakan suatu hal yang sama? Silahkan anda coba selidiki.

```
print(jumlahKuadrat(b=int(input()),a=int(input())))
print(jumlahKuadrat(int(input()),int(input())))
```

Jika suatu fungsi memiliki parameter yang bersifat opsional, maka dapat diset sesuai keperluan. Umumnya diset bernilai nol. Artinya ketika argument tidak diisi, maka argument akan diisi dengan nilai nol, seperti contoh pada Gambar 5.5. Pada pemanggilan pertama fungsi jumlahkan, argument yang dikirimkan ada tiga, sesuai dengan parameternya, sehingga nilai bil3 adalah 5. Tetapi pada pemanggilan kedua argument yang dikirimkan hanya dua, sehingga pada pemanggilan kedua ini nilai bil3 adalah 0.

```
def jumlahkan(bil1, bil2, bil3=0):
    return (bil1+bil2+bil3)

#Program Utama
print(jumlahkan(1,4,5))
print(jumlahkan(1,4))
```

Gambar 5. 5 Contoh parameter optional

Perlu diingat untuk menempatkan definisi fungsi sebelum fungsi utama. Dengan ini otomatis compiler mengetahui bahwa program utama menggunakan fungsi yang didefinisikan. Dalam implementasi, jika

program berukuran besar biasanya akan dibagi menjadi beberapa fungsi yang kecil. Kadang suatu fungsi memerlukan hasil dari fungsi lain yang sudah didefinisikan secara terpisah, untuk itu diperlukan pemanggilan fungsi oleh fungsi yang lain. Pemanggilan fungsi oleh suatu fungsi merupakan suatu hal yang diijinkan. Gambar 5.6 memperlihatkan contoh pemanggilan antar fungsi, yaitu pemanggilan fungsi jumlahkan oleh fungsi kalikan.

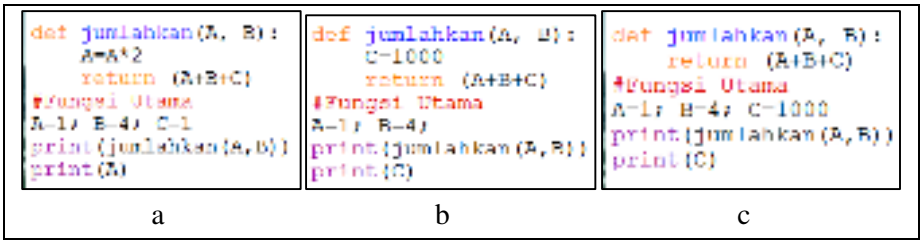
```
def kalikan(a,b):
    x=jumlahkan(a,b)
    return (2*x)

def jumlahkan(A, B):
    return (A+B)

#Fungsi Utama
print(kalikan(1,4))
```

Gambar 5. 6 Pemanggilan fungsi oleh fungsi lain

5.5. Variabel Lokal dan Variabel Global

Parameter merupakan variabel tujuan yang khusus didefinisikan di dalam tanda kurung dari definisi fungsi. Tujuannya adalah untuk menyimpan informasi yang dikirimkan oleh argumen, yang terdaftar di dalam tanda kurung dari pemanggilan fungsi. Biasanya ketika informasi dikirimkan ke suatu fungsi, maka parameter akan menerima salinan dari nilai yang dikirimkan. Jika nilai parameter diubah di dalam suatu fungsi, maka hal itu tidak akan berpengaruh pada argumen yang aslinya. Perhatikan Gambar 5.7.a. Pada program ini nilai variabel A dikirimkan sebagai argument pemanggilan fungsi **`jumlahkan`**. Kemudian pada fungsi **`jumlahkan`** nilai variabel A diubah menjadi dua kali nilai argument yang diterimanya. Penugasan ini tidak akan berpengaruh terhadap nilai dari variabel A pada fungsi utama sebagai pemanggil. Hal ini terjadi karena variabel A pada fungsi **`jumlahkan`**, hanya berisi salinan dari nilai variabel A pada fungsi utama. Sehingga pada saat perintah print(A) muncul setelah pemanggilan fungsi, yang ditampilkan di layar adalah nilai variabel A semula, yaitu 1. Hanya salinannya saja yang diubah, bukan yang asli. Fungsi **`jumlahkan`** tidak memiliki hak akses terhadap variabel A dan B pada fungsi utama.

Pengiriman argument berupa variabel merupakan pembatasan hak ases fungsi yang dipanggil terhadap variabel tersebut.



Gambar 5. 7 Variabel lokal dan variabel global

Pada program Gambar 5.7.b, fungsi **`jumlahkan`** memiliki variabel C yang diset bernilai 1000. Variabel ini merupakan variabel milik fungsi **`jumlahkan`**, fungsi lain tidak memiliki hak akses secara langsung. Fungsi utama tidak memiliki variabel C, sehingga saat pada fungsi ini ada penugasan print(C), akan muncul pesan kesalahan.

Variabel-variabel yang didefinisikan di dalam suatu fungsi penggunaannya lokal untuk fungsi tersebut. Mereka disembunyikan dari pernyataan di fungsi lain, dan biasanya tidak dapat mengaksesnya. Variabel lokal hanya ada saat fungsi yang mendefinisikannya sedang dieksekusi. Ketika fungsi dimulai, variabel parameternya dan variabel lokal apa pun yang didefinisikannya dibuat dalam memori, dan ketika fungsi berakhir, mereka dihancurkan. Ini berarti bahwa nilai yang disimpan dalam parameter fungsi ataupun variabel lokalnya akan hilang setelah pemanggilan fungsi selesai.

Perhatikan kembali Gambar 5.7.c. Pada program ini, fungsi **`jumlahkan`** hanya memiliki variabel local A dan B. Tetapi dia dapat menghitung nilai (A+B+C). Hal ini terjadi karena fungsi utama memiliki variabel C, sehingga nilai variabel C inilah yang digunakan oleh fungsi **`jumlahkan.`** Variabel C pada contoh ini merupakan suatu variabel global, yaitu variabel yang dapat diakses oleh semua fungsi. Pada program 5.7.c, fungsi utama memiliki A dan B sebagai variabel local, dan C sebagai variabel global. Sedangkan fungsi **`jumlahkan`** hanya memiliki variabel lokal A dan B.

5.6. Fungsi Rekursif

Suatu fungsi dapat didefinisikan untuk melakukan pemanggilan terhadap fungsi lain maupun dirinya sendiri. Fungsi yang didefinisikan untuk memanggil dirinya sendiri disebut fungsi rekursif. Perhatikan contoh berikut pada Gambar 5.8.

```
def turunkan(N):
    if N <= 0:
        print("Selesai")
    else:
        print(N)
        turunkan(N-1)

#Fungsi utama
turunkan(3)
```

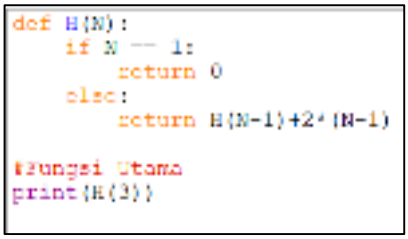
Gambar 5. 8 Contoh fungsi rekursif

Fungsi **`turunkan`** merupakan fungsi rekursif. Ketika nial N kurang dari atau sama dengan nol, fungsi tersebut akan menampilkan pesan "Sudah habis", jika tidak dia akan memanggil fungsi **`turunkan`** (dirinya sendiri), dengan argument N-1. Apa yang terjadi jika kita memanggil fungsi yang seperti ini? Ilustrasinya dapat dilihat pada Tabel 5.1.

Tabel 5. 1 Ilustrasi proses rekuren fungsi turunkan

Proses	Hasil
Mulai masuk fungsi dengan argument 3. Karena lebih dari 0, maka tampilkan nilai ke layar, kemudian panggil fungsi dengan argument 2. Masuk ke fungsi dengan argument 2. Karena lebih dari 0, maka tampilkan nilai ke layar, kemudian panggil fungsi dengan argument 1. Masuk ke fungsi dengan argument 1. Karena lebih dari 0, maka tampilkan nilai ke layar, kemudian panggil fungsi dengan argument 0. Masuk ke fungsi dengan argument 0. Karena sama dengan 0, maka tampilkan pesan ke layar, dan *return*. Fungsi dengan N=1, *return* Fungsi dengan N=2, *return* Fungsi dengan N=3, *return* Kembali ke fungsi utama. maka akan muncul hasil akhir. 3 2 1 Selesai	3 2 1 Selesai

Struktur fungsi yang didefinisikan secara rekursif akan sangat bermanfaat untuk menyelesaikan masalah. Program pada Gambar 5.7 dapat digunakan misalnya untuk menampilkan waktu pengaturan pada stopwatch. Fungsi yang didefinisikan secara rekursif, juga dapat mengembalikan suatu nilai. Misalkan didefinisikan suatu rekursif, H(n) = H(n-1)+n-1, H(1)=0. Maka program pada Gambar 5.9 merupakan kode program untuk fungsi H.



Gambar 5. 9 Contoh fungsi rekursif dengan nilai balikan

Fungsi rekursif yang bernama H akan mengembalikan nilai nol, jika nilai argument N sama dengan satu. Jika tidak fungsi akan memanggil fungsi H (dirinya sendiri) dengan argument N-1. Apa yang terjadi pada program 5.9, dapat dipelajari pada Tabel 5.2.

Tabel 5. 2 Ilustrasi proses rekuren fungsi H

Proses
Mulai masuk fungsi H dengan argument 4. Karena tidak sama dengan 1, maka panggil fungsi H dengan argument 3. Mulai masuk fungsi H dengan argument 3. Karena tidak sama dengan 1, maka panggil fungsi H dengan argument 2. Masuk ke fungsi H dengan argument 2. Karena tidak sama dengan 1, maka panggil fungsi H dengan argument 1. Masuk ke fungsi H dengan argument 1. Karena sama dengan 1, maka return 0. Fungsi H dengan N = 2 dan balikan H(N-1)=0. *return* (0+2-1)=1. Fungsi dengan N=3 dan balikan H(N-1)=1, *return* (1+3-1)=3 Fungsi dengan N=4 balikan H(N-1)=3, *return* (3+4-1)=6 Kembali ke fungsi utama. maka akan muncul hasil akhir. 6

5.7. Kesimpulan

Hal yang dapat disimpulkan dari pembahasan ini adalah:

1. Suatu fungsi merupakan bagian program yang berisi kumpulan statements untuk melaksanakan tugas tertentu.
2. Stuktur umum fungsi
   def function-name(parameter1, parameter2,…):
       statement-1
       statement-2
       ……..
       statement-n
3. Bila suatu fungsi memiliki balikan suatu nilai diakhiri dengan statement *return.*
4. Format umum pemanggilan fungsi adalah
   function-name(argumen1, argumen2,…)
5. Parameter dan argument dipasangkan menurut urutan
6. Variabel yang ada didalam suatu fungsi merupakan variabel local, tidak dapat diakses dari luar fungsi.
7. Variabel global merupakan variabel yang dapat digunakan oleh semua fungsi.
8. Fungsi rekursif adalah fungsi yang didefinisikan dapat memanggil dirinya sendiri.

C. Latihan 5

1. Apakah yang berikut merupakan header fungsi atau pemanggilan fungsi?
   hitungTotal()
2. Apakah yang berikut merupakan header fungsi atau pemanggilan fungsi?
   cetakHasil()
3. Apa output program berikut, jika pengguna memasukkan nilai 10?

```
def func1():
    print("A", end=' ')
def func2():
    print("B", end=' ')

#main
nilai= int(input())    #masukkan 10
if(nilai < 10):
        func1()
        func2()
else:
        func2()
        func1()
```

4. Amati berikut ini mana yang merupakan header fungsi dan pemanggilan fungsi.

```
Hitung(bilangan)
def faktorkan(A):
jumlah(3, A, 8)
```

5. Apa output dari program berikut?

```
def func1(a,b):
    print("fungsi1\t", a," ",b)
    a = 0.0; b = 1
    print("fungsi2\t", a," ",b)

main()
x = 0
y = 2.5
print("main1\t", x," ", y)
func1(y, x)
print("main2\t", x," ", y)
```

6. Apa output dari program berikut?

```
def func1():
    print("A", end=' ')
def func2():
    print("B", end=' ')

#main
for nilai in range(1,6):
    if(nilai % 3==0):
        func1()
        func2()
    else:
        func2()
        func1()
```

7. Apa output dari program berikut?

```
def fung(A, B):
    return (A+B)


#main
n = 1
X=fung(2, fung(n,3))
print(X)
```

8. Apa output dari program berikut?

```
def fung(A, B, C=1):
    return (A+B)/C


#main
a=1; b=2
X=fung(a,b)
print(X)
```

9. Apa output dari program berikut?

```
def fung(A):
    if(A>1):
        return (fung(A-1)+3)
    else:
        return 1

#main
n = 3
print(fung(n))
```

10. Bagian dari definisi fungsi yang memuat nama fungsi, dan daftar parameter disebut .....
11. Nilai yang dikirimkan ke suatu fungsi disebut _____.
12. Jika fungsi fungsiKu memiliki header berikut:
    def fungsiKu(Nilai):
    Tuliskan pernyataan pemanggilan fungsi tersebut dengan argumen 5
    _____
13. Variabel khusus tempat menyimpan salinan nilai dari argumen fungsi disebut _____.
14. Variabel _____ didefinisikan di dalam fungsi dan tidak dapat diakses di luar fungsi.
15. Apa bedanya argumen dengan variabel parameter?

16. Ketika suatu fungsi menerima argumen lebih dari satu, apakah urutan argumen yang dikirimkan harus diperhatikan?
17. Pernyataan berikut memanggil fungsi bernama bagi3, yang mengembalikan suatu nilai sepertiga dari nilai argumen yang diterimanya. Tuliskan definisi fungsi bagi3 tersebut.

```
hasil = bagi3(bilangan)
```

18. Suatu program memuat fungsi berikut.

```
def pangkat4(bil):
    return bil * bil * bil * bil
```

Tuliskan pernyataan pemanggilan fungsi dengan argumen yang dikirim 3, dan menyimpan hasilnya pada variabel fx.

19. Tulis suatu fungsi (hanya fungsi, bukan program lengkap), yang menerima argumen satu bilangan bulat. Fungsi akan menampilkan hasil perkalian argumen yang diterimanya dengan 5.
20. Suatu program memuat fungsi berikut

```
def tampilkan(bil1, bil2, bil3):
 print("Nilainya adalah: ", bil1, " ,", bil2, ", ",bil3)
```

Tulis pernyataan yang memanggil fungsi tersebut, dan sebagai argument kirimkan variabel a, b, c.

21. Tulis suatu fungsi yang akan menerima dua argumen berupa bilangan. Fungsi akan mengembalikan nilai f(x), jika diketahui.

$$f(x) = \frac{a+b}{ab}, ab \neq 0$$

22. Tuliskan program untuk mengitung nilai z, jika diketahui

$$z(x) = \begin{cases} \sqrt{x} + x + 1, jika\ x > 0 \\ x^3 - x + 1, jika\ x < 0 \\ 0, \qquad\quad jika\ x = 0 \end{cases}$$

23. Buat program untuk menghitung volum Tabung yang diketahui ukuran tinggi dan jari-jari alasnya. Program paling tidak memiliki dua fungsi, yaitu fungsi untuk menghitung volume Tabung, dan fungsi untuk menghitung luas lingkaran.

    Catatan:

    rumus volum tabung dengan tinggi t, dan jari-jari r adalah: $V = \pi r^2 t$

    rumus luas lingkaran dengan jari-jari r: $L = \pi r^2$
24. Program berikut akan menanyakan kepada pengguna berapa jam lamanya bekerja dan berapa upah per jamnya. Program akan

menghitung dan menampilkan total upah yang akan diterima. Fungsi **tampilkanRupiah** yang akan anda tulis, akan menampilkan ke layar total upah yang akan diterima.

```
#Fungsi utama
jamkerja =float(input("Berapa jam anda bekerja? "))
upah = int(input("Berapa upah anda per jam? "))
totalUpah = jamkerja * upah
tampilkanRupiah (totalUpah)
print("SELESAI")
```

Lengkapi program di atas dengan definisi fungsi **tampilkanRupiah** yang sesuai. Fungsi memiliki satu parameter. Buat tampilan output dengan kreatif, dan tampilkan nilai float dalam format dua angka desimal.

BAB 6 PENGULANGAN (LOOPING)

A. Tujuan Pembelajaran

Tujuan yang ingin dicapai dalam pembelajaran ini adalah

1. Mahasiswa dapat memberikan contoh masalah terkait konsep pengulangan dengan benar.
2. Diberikan suatu masalah, mahasiswa dapat menghasilkan program untuk menyelesaikan masalah tersebut melibatkan penggunaan while dalam Python, dengan jujur dan kreatif.
3. Diberikan suatu masalah, mahasiswa dapat menghasilkan program untuk menyelesaikan masalah tersebut melibatkan penggunaan while bersarang pada Python dengan jujur dan kreatif.
4. Diberikan suatu program terkait penggunaan while, mahasiswa dapat menentukan output dari program tersebut secara manual dengan benar.
5. Diberikan suatu masalah, mahasiswa dapat menghasilkan program untuk menyelesaikan masalah tersebut melibatkan penggunaan for pada Python dengan jujur dan kreatif.
6. Diberikan suatu program terkait penggunaan for, mahasiswa menentukan output suatu pogram tersebut secara manual dengan benar.
7. Diberikan suatu masalah, mahasiswa secara berkelompok dapat menghasilkan program untuk menyelesaikan masalah tersebut melibatkan penggunaan for bersarang pada Python dengan jujur dan kreatif, serta mempresentasikannya dengan bertanggung jawab.

B. Uraian Materi

Loop adalah struktur kontrol yang secara berulang mengeksekusi suatu himpunan pernyataan. Artinya program akan melaksanakan satu atau lebih pernyataan berkali-kali, sampai kondisi tertentu dipenuhi. Untuk mencapai kondisi tertentu ini diperlukan suatu pengatur. Sebagai pengatur dalam pengulangan biasanya digunakan operator penambahan atau pengurangan. Menambah nilai berarti meningkatkannya, dan mengurangi nilai berarti

menurunkannya. Dalam contoh di bawah ini, nilai variabel bilangan akan bertambah 10.

```
bilangan = bilangan + 10
```

Perlu diingat lagi bahwa tanda "=" dalam kode Python berbeda maknanya dengan tanda "=" di dalam matematika. Seperti dijelaskan pada Bab 2, tanda "=" di dalam Python menujukkan suatu penugasan. Sehingga jika asalnya variabel bilangan berisi nilai 3, maka setelah perintah di atas dilaksanakan variabel bilangan akan diisi dengan nilai 13. Perintah berikut akan menyebabkan nilai variabel banyakPeserta dikurangi 3.

```
banyakPeserta = banyakPeserta – 3
```

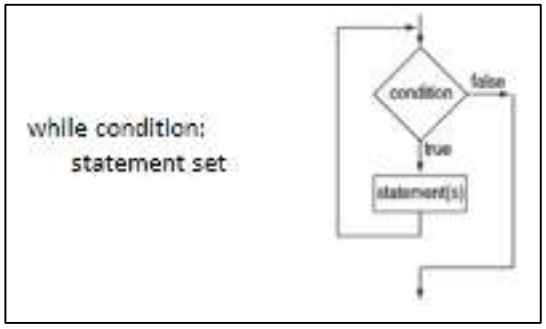
Pernyataan-pernyataan di atas dapat ditulis dengan cara yang lebih singkat tanpa mengubah hasil, yaitu menjadi

```
Bilangan += 10
banyakPeserta -= 3
```

Ada beberapa struktur pengulangan yang disediakan oleh Python. Masing-masing struktur memiliki cara kerja masing-masing. Struktur pengulangan tersbut adalah pengulangan dengan kata kunci *while* dan dengan kata kunci *for*.

6.1. Pengulangan dengan while

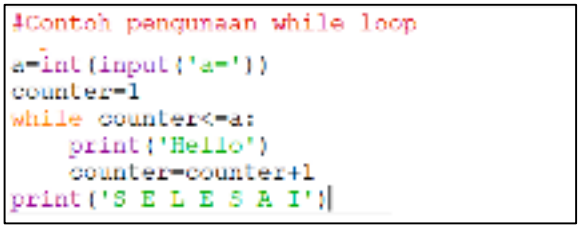
Salah satu struktur yang disediakan Python untuk melakukan pengulangan adalah struktur *while*. *While loop* merupakan struktur pengulangan yang memiliki dua bagian penting, yaitu: (1) ekspresi yang diuji untuk nilai benar atau salah, dan (2) himpunan pernyataan atau blok atau suite yang diulang selama ekspresi bernilai benar. Gambar 6.1 menunjukkan format umum while loop dan bagan alur yang secara visual menggambarkan cara kerjanya.



Gambar 6. 1 Struktur while loop

Baris pertama, disebut header loop, terdiri dari kata kunci **`while`** diikuti oleh kondisi yang akan diuji dan diakhiri dengan tanda titik dua. Kondisi merupakan ekspresi apa pun yang dapat dievaluasi sebagai benar atau salah. Selanjutnya adalah tubuh loop atau blok/ suite. Bagian ini memuat beberapa statement dalam Python.

Langkah pertama dalam kerja while loop adalah menguji nilai dari ekspresi kondisi, dan jika bernilai benar, setiap pernyataan dalam tubuh loop akan dieksekusi. Kemudian, kondisinya diuji lagi. Jika nilainya masih benar, setiap pernyataan dieksekusi lagi. Siklus ini berulang hingga kondisinya bernilai salah. Perhatikan bahwa, seperti halnya pada pernyataan if, setiap pernyataan dalam tubuh while loop yang dieksekusi secara kondisional ditulis secara indent (menjorok ke dalam). Ini karena while loop tidak lengkap tanpa pernyataan yang mengikutinya. Seperti diperlihatkan pada diagram Gambar 6.1, pada dasarnya while loop berfungsi seperti pernyataan if yang dapat dieksekusi secara berulang-ulang sampai suatu kondisi tertentu tidak terpenuhi. Gambar 6.2 memperlihatkan contoh penggunaan while loop untuk menampilkan tulisan Hello berulang-ulang sesuai dengan inputan dari pengguna.



Gambar 6. 2 Contoh penggunaan while loop

Pada baris ke-3 variabel counter diinisialisasi dengan nilai 1. Pada baris ke-4, while loop dimulai dengan pernyataan:

```
while counter <= a:
```

Pernyataan ini menguji variabel counter untuk menentukan apakah nilainya kurang dari atau sama dengan nilai variabel a. Jika bernilai benar, pernyataan dalam tubuh loop (baris ke-5 dan ke-6) dieksekusi:

```
print('Hello')
counter=counter+1
```

Pernyataan di baris ke-5 mencetak kata "Hello". Pernyataan pada baris ke-6 menambahkan satu ke varibel counter, sehingga nilainya menjadi 2. Ini adalah pernyataan terakhir dalam tubuh loop, sehingga berikutnya loop dimulai lagi dari awal. Diuji lagi apakah variabel counter nilainya kurang dari atau sama dengan nilai variabel a, dan jika nilainya masih benar, pernyataan dalam tubuh loop dieksekusi lagi. Siklus ini berulang hingga counter bernilai lebih dari nilai variabel a, artinya ekspresi kondisi bernilai salah. Setiap satu kali eksekusi tubuh loop dikenal sebagai iterasi. Jadi iterasi adalah suatu prosedur dimana beberapa pekerjaan dilakukan berulang kali. Pada struktur while loop, iterasi akan berhenti jika kondisi bernilai salah, selanjutnya program akan mengeksekusi pernyataan berikutnya. Tabel 6.1 memperlihatkan proses yang terjadi pada while loop yang diperlihatkan pada contoh program Gambar 6.2, jika a diberi nilai 4.

Tabel 6.1 Cara kerja while loop

Iterasi	a	Counter	counter <=a	print('Hello')	counter=counter+1
1	4	1	Benar	Hello	counter=1+1=2
2	4	2	Benar	Hello	counter=2+1=3
3	4	3	Benar	Hello	counter=3+1=4
4	4	4	Benar	Hello	counter=4+1=5
5	4	5	Salah	Loop berhenti	

Variabel yang mengontrol banyaknya eksekusi yang diperlukan loop disebut sebagai variabel kontrol loop. Dalam contoh 6.2, counter merupakan variabel kontrol loop. While loop dikenal sebagai loop pretest, yang artinya loop akan menguji ekspresi sebelum setiap iterasi. Perhatikan definisi variabel counter di baris ke-3,

```
counter = 1
```

Pada pernyataan ini variabel counter diinisialisasi dengan nilai 1. Jika variabel a diberi nilai 0, loop tidak akan pernah dieksekusi. Karakteristik penting dari while loop adalah bahwa loop tidak akan pernah dieksekusi jika mengujian ekspresi kondisi di awal bernilai salah. Jadi jika ingin memastikan apakah while loop akan dieksekusi pertama kalinya, data yang relevan harus diinisialisasi sedemikian rupa sehingga pengujian ekspresi kondisi dimulai dengan nilai benar. Untuk lebih jelas pelajari contoh program pada Gambar 6.3.

```
#Contoh pengunaan while loop
a=5
counter=1
while counter>=a:
    print('Hello')
    counter=counter+1
print('S E L E S A I')
```

Gambar 6. 3 While loop yang tidak pernah dikerjakan

Agar iterasi dapat berakhir, loop harus memuat cara untuk mengakhiri pengulangan. Ini berarti bahwa sesuatu di dalam loop pada akhirnya harus membuat ekspresi kondisi bernilai salah. Loop dalam contoh program 6.2 berhenti ketika ekspresi counter <= a bernilai salah. Jika loop tidak memiliki cara untuk berhenti, maka loop tersebut disebut loop tak terbatas. Karena loop terus berulang tak terbatas sampai program terganggu. Berikut ini sebuah contoh:

```
counter = 1
while counter <= 5:
        print("Hello    ")
```

Program ini merupakan loop tak terbatas karena tidak mengandung pernyataan yang mengubah nilai variabel counter. Sehingga setiap kali ekspresi counter <= 5 diuji, variable counter akan tetap berisi nilai 1. Jadi

jangan lupa jika menulis while loop harus memuat statement untuk mengubah nilai dari variabel pengontrol. Perhatikan program berikut.

```
counter = 1
while counter <= 5:
      print("Hello   ")
counter = counter+1
```

Pada program ini, terdapat statement yang mengubah nilai dari variabel pengontrol, yaitu counter = counter+1, tetapi statement ini tidak ditulis menjorok (indent). Berarti statement ini berada di luar blok dari loop. Loop ini akan dieksekusi tak terbatas kali, karena di dalam blok dari loop tidak memuat statement untuk mengubah nilai variabel counter.

Selama ini ketika suatu program dieksekusi, jika akan mencoba eksekusi lagi dengan nilai input yang berbeda, maka kita harus mengulang menekan F5. Dengan menggunakan while loop, pengguna dapat mengulang menjalankan suatu program sesuai keperluan. Contoh pada Gambar 6.4 memungkinkan pengguna untuk menjalankan program beberapa kali sesuai dengan keperluan.

```
print('Menghitung Luas Daerah Persegi Panjang')
ulang='y'
while ulang=='y' or ulang=='Y':
    p=int(input('panjang='))
    l=int(input('lebar='))
    satuan=input('satuan=')
    L=p*l
    print('luas={} {} persegi'.format(L,satuan))
    ulang=input('Ingin mengulang (y/t)?');
print('SELESAI')
```

Gambar 6. 4 Contoh loop untuk mengulang menjalankan program

Loop juga dapat digunakan untuk memvalidasi inputan yang dimasukkan oleh pengguna apakah sesuai atau tidak, seperti diperlihatkan pada Gambar 6.5.

```
print( "Masukkan bilangan pada  range 1 - 100: ")
bilangan=int(input())
while (bilangan < 1) or (bilangan > 100):
    print("ERROR: Masukkan bilangan pada  range 1 - 100:")
    bilangan=int(input())
```

Gambar 6. 5 Contoh lain penggunaan loop

Program ini akan memaksa pengguna memasukkan nilai pada interval 1-100. Jika nilai yang diinputkan di luar interval tersebut akan ke luar pesan kesalahan, dan pengguna diminta untuk memasukkan bilangan yang lain. Dapat juga digunakan statement **break**, sehingga jika suatu kondisi tidak terpenuhi, maka program akan ke luar dari loop, seperti contoh pada Gambar 6.6.

```
bilangan= 6
while (bilangan < 10):
    if(bilangan%3==2):
        print(bilangan,' bersisa 2 jika dibagi 3')
        break
    bilangan+=1
```

Gambar 6. 6 Contoh penggunaan keyword Break pada loop

Pada program 6.6, program akan ke luar dari while, jika sisa pembagian bilangan dengan 3 bernilai 2. Jika kita ingin meloncati (skip) suatu kondisi tertentu, dapat digunakan statement **continue**, seperti diperlihatkan pada Gambar 6.7.

```
i = -3
while i < 4:
  i+=1
  if i == 0:
    print('pembagian dengan 0')
    continue
  A = 5/i
  print(i,'\t', A)
```

Gambar 6. 7 Contoh penggunaan keyword continue

Program pada Gambar 6.7 akan menampilkan nilai variabel i dan A, untuk i mulai -2 sampai dengan 4 kecuali ketika variabel i bernilai nol. Karena pembagian dengan suatu bilangan nol akan menyebabkan program berhenti dengan pesan kesalahan.

Terkadang penting bagi program untuk melacak banyaknya iterasi yang dilakukan loop. Bagaimana menampilkan banyaknya iterasi yang dilakukan suatu loop?

Latihan

1. Berapa kali kata "Corona\n" akan ditampilkan pada masing-masing potongan program berikut?

```
a) count = 1
   while (count < 5):
          print( "Semua gara-gara ", end=' ')
          print("Corona")
          count=count+1
b) count =10
   while (count < 5):
          print( "Semua gara-gara ", end=' ')
          print("Corona")
          count=count+1
c) count = 1
   while (count < 5):
          print( "Semua gara-gara ", end=' ')
          print("Corona")
          count=count-1
```

2. Pada soal nomor 1 a), variabel apa yang menjadi pengontrol loop?
3. Tuliskan loop untuk memvalidasi input dari pengguna yang diminta untuk melakukan yang berikut:
   a) Memasukkan pilihan menu 1,2,3, atau 4.
   b) Masukkan 'Y', 'y', 'N', atau 'n'.

4. Tentukan output dari program berikut

```
M=1; A = 2; y=1;
while(y<A):
    M=M*y+2
    y = y + 0.5
    print(M)

print(y, "\t", A, "\t", M)
```

5. Tentukan output dari program berikut, jelaskan fungsi dari pernyataan **break**

```
n=6
while n>0:
    if n==3:
        break
    print(n)
    n-=1
print("end loop")
```

6.2. For Statement

Pengulangan/loop merupakan pengulangan pretest yang mengkombinasikan inisialisasi, pegujian dan apdate penghitung pada satu header. Secara umum terdapat dua jenis pengulangan: conditional loops and count-controlled loops. Suatu conditional loop dieksekusi sepanjang kondisi tertentu dipenuhi. Contoh, validasi input terus dilakukan sepanjang nilai input invalid seperti pada contoh pengulangan dengan while. Conditional loop, merupakan pengulangan yang digunakan ketika banyak kali pengulagan atau iterasi tidak diketahui. Jika berapa kali iterasi harus dilakukan sudah diketahui, maka pengulangan yang demikian disebut count-controlled loop.

Contoh kasus untuk count-controlled loop adalah, jika pengulangan meminta pengguna untuk memasukkan jumlah penjualan untuk setiap bulan dalam setahun, maka iterasi akan dilakukan 12 kali. Penghitung pengulangan menghitung sampai 12 dan meminta pengguna memasukkan nilai jumlah penjualan setiap waktu. Penghitung pengulangan harus melakukan tiga hal:

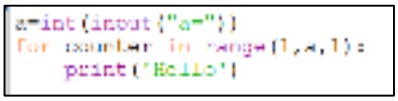
1. Inisialisasi variabel penghitung menjadi nilai awal.

2. Uji variabel penghitung dengan membandingkannya dengan nilai akhir. Jika variabel penghitung sama dengan nilai akhir, pengulangan berhenti.
3. update variable counter sepanjang iterasi. Biasanya dilakukan dengan menaikkan satu-satu variable counter.

Struktur pengulangan terkontrol yang dapat digunakan dalam Python adalah statement for. Pengulangan for khusus didisain untuk melakukan inisialisasi, menguji, dan update variable pengontrol dalam header. Secara umum format pengulangan dengan for adalah sebagai berikut.

```
for k in range(start, end, update):
        statement set
statement out of loop
```

Baris pertama pada *for loop* disebut header. Kata kunci **for** diikuti dengan variabel pengontrol (k) dan kata kunci **in,** kemudian fungsi **range**(). Fungsi range menunjukkan interval pengulangan dari **start** sampai **end**, dengan perubahan variabel pengontrol menggunakan **update**. Seperti biasa header pada Python selalu diakhiri dengan tanda titik dua. Gambar 6.8 memperlihatkan contoh penggunaan for loop hasil modifikasi dari program pada Gambar 6.2.



Gambar 6. 8 Contoh penggunaan for loop

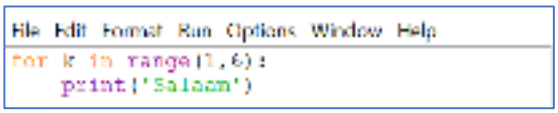
Pada program 6.8, nilai counter dimulai dari 1, dan nilainya naik satu-satu. Nilai ini akan dievaluasi apakah sama dengan nilai variabel a atau tidak. Jika tidak, maka program akan menampilkan string "Hello". jika ya maka program akan ke luar dari struktur *loop.* Tabel 6.2 memperlihatkan cara kerja for loop pada program 6.8 dengan nilai a dimisalkan 4.

Tabel 6.2 Cara kerja for loop

Iterasi	**a**	**counter**	**counter ==a**	**print('Hello')**	**update**
1	4	1	Salah	Hello	counter = 1+1 = 2
2	4	2	Salah	Hello	counter = 2+1 = 3
3	4	3	Salah	Hello	counter = 3+1 = 4
4	4	4	Benar	Loop berhenti	

Berdasarkan Tabel 6.2, dapat dilihat bahwa nilai counter ketika ke luar dari struktur for loop adalah 4, dengan output tiga kali menampilkan string “Hello”. Ini berbeda dengan hasil penggunaan while loop pada Tabel 6.1. Bagaimana caranya agar kedua program memiliki output yang sama?, silahkan anda ekplorasi.

Standar dari update nilai counter pada for loop adalah naik satu-satu, sehingga jika tidak dituliskan update pada argument fungsi range, system otomatis akan melakukan update terhadap counter dengan menaikkan nilainya satu-satu. Gambar 6.9 berikut memperlihatkan contoh penggunaan for loop untuk menampilkan tulisan Salaam sebanyak lima kali, tanpa mengirimkan argument untuk fungsi *range*.

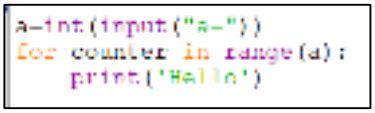


Gambar 6. 9 Contoh statement for

Pada contoh Gambar 6.9, sebagai variabel pengontrol adalah k. Update variabel pengontrol tidak dituliskan karena nilainya satu. Artinya nilai k akan naik satu-satu atau k=k+1. Seperti halnya pada pengulangan dengan while, pada pengulangan dengan for pun batas awal dan batas akhir dapat ditentukan secara bebas, misalnya sebagai inputan dari pengguna. Jika nilai k mulai dari A sampai kurang dari B, dan perubahannya naik dengan penambahan satu, maka header untuk for dapat ditulis menjadi

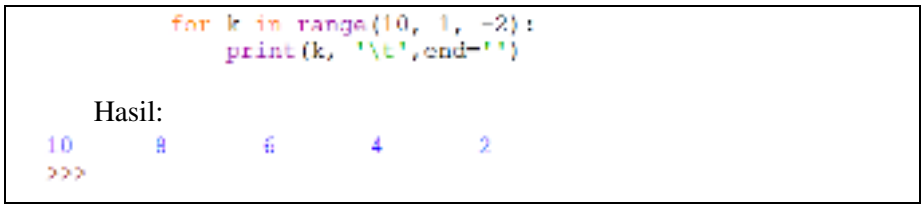
```
for k in range(A, B):
```

Jika pada header pemanggilan fungsi *range* hanya mengirimkan satu argument saja, maka otomatis nilai awal dari variabel penentu akan diset 0, dan perubahannya naik dengan penambahan satu. Hal ini dapat dilihat pada Gambar 6.10.



Gambar 6. 10 Contoh header loop dengan satu argumen untuk fungsi range

Pada program 6.10, string "Hello" akan muncul sebanyak a kali, karena nilai counter dimulai dari 0 sampai kurang dari a, dan nilai a naik dengan penambahan 1. Perubahan nilai variable pengontrol dalam for loop tidak harus naik dengan penambahan satu. Nilai variable pengontrol dapat naik ataupun turun berapapun sesuai dengan keperluan. Gambar 6.10 adalah contoh penggunaan for loop dimana nilai variabel pengontrol turun dengan pengurangan dua. Karena nilainya turun, tentu saja nilai awal dari variabel penentu harus lebih dari nilai akhirnya.



Gambar 6. 11 Contoh program dan output dari statement for

Seperti halnya pada pengulangan dengan while, dalam pengulangan dengan for pun, statement break dan continue dapat diterapkan. Silahkan coba contoh program penggunaan break dan continue pada pengulangan while, anda ubah menjadi menggunakan pengulangan for.

6.3. Nested Loop (Loop Bersarang)

Seperti halnya penggunaan struktur percabangan, struktur pengulangan juga dapat dirancang secara bersarang. Jadi struktur pengulangan di dalam struktur pengulangan yang lain. Sebelumnya kita sudah mempejari bahwa untuk menampilkan keluaran berikut

10 8 6 4 2

Dapat dilakukan diantaranya dengan menggunakan for loop seperti pada contoh Gambar 6.9. Lalu bagaimana jika kita ingin tampilan hasilnya seperti berikut?

```
10    8    6    4    2
10    8    6    4    2
10    8    6    4    2
10    8    6    4    2
```

Untuk menampilkannya menjadi empat kali, artinya code pada Gambar 6.9 harus dilakukan sebanyak empat kali. Berarti terjadi pengulangan empat kali, terhadap loop pada Gambar 6.9. Untuk ini dapat digunakan digunakan statement for ataupun while. Gambar 6.10 berikut menunjukkan penggunaan statement for untuk masalah tersebut.

```
for i in range (1, 7):
    for k in range(10, 1, -2):
        print(k, '\t',end='')
    print()
```

Gambar 6. 12 Contoh statement for bersarang

Penulisan loop seperti ini disebut nested loop atau loop bersarang. Artinya di dalam pengulangan ada lagi pengulangan. Dalam suatu loop bersarang, pengulangan yang paling dalam akan diselesaikan lebih dahulu. Jadi arah penyelesaian adalah dari loop yang paling dalam bergerak sampai loop terluar. Perhatikan contoh pada Gambar 6.11.

```
for j in range (1, 3):
    print('j=', j)
    for i in range(0, 3):
        print('  i=',i)
        for k in range(1, 5):
            print('\tk=', k,end='')
        print()
```

Gambar 6. 13 Contoh statement for bersarang dua kali

Pada loop bersarang Gambar 6.11, tulisan "k=" akan muncul sebanyak 4×3×2 = 24 kali. Sedangkan tulisan "i=" akan muncul sebanyak 3×2= 6 kali, dan tulisan "j=" akan muncul 2 kali. Hasil program lengkap akan nampak seperti pada Gambar 6.12.

```
j= 1
  i= 0
        k= 1    k= 2    k= 3    k= 4
  i= 1
        k= 1    k= 2    k= 3    k= 4
  i= 2
        k= 1    k= 2    k= 3    k= 4
j= 2
  i= 0
        k= 1    k= 2    k= 3    k= 4
  i= 1
        k= 1    k= 2    k= 3    k= 4
  i= 2
        k= 1    k= 2    k= 3    k= 4
```

Gambar 6. 14 Contoh output program Gambar 6.11

Program berikut merupakan contoh loop bersarang lainnya. Loop dengan variable pengontrol i, memiliki dua loop di dalamnya, yaitu loop dengan variable pengontrol k, dan loop dengan variable pengontrol j.

```
t = int (input())
for i in range(1,t+1):
    for k in range(t-i, 0, -1):
        print("  ", end="")     #2 spasi
    for j in range(i,0, -1):
        print('$ ', end='')
    print()
```

Gambar 6. 15 Contoh loop bersarang lainnya

Berbeda dengan contoh sebelumnya, dalam contoh ini loop k dan loop j merupakan loop yang sekuensial. Keduanya setara, karena mempunyai jarak indentasi yang sama. Untuk setiap nilai i, loop k akan diselesaikan lebih dahulu, baru menyelesaikan loop j. Cobalah program tersebut gambar apa yang terbentuk?

6.4. Kesimpulan

Berdasarkan yang telah diuraikan, dapat disimpulkan bahwa

1. Format umum struktur while loop adalah

```
counter initialization
while condition:
      statement set
      counter update
statement out of while
```

2. Format umum struktur for loop

```
for k in range(start, end, update):
       statement set
statement out of loop
```

C. Latihan 6

1. Tentukan output dari program berikut, jelaskan fungsi dari pernyataan

```
exit()
  n=6
  while n>0:
      if n==3:
          exit()
      print(n)
      n-=1
  print("end loop")
```

2. Tuliskan dalam struktur for, untuk x mulai dari 1 sampai 10 dengan kenaikan satu-satu, jika x bilangan genap tampilkan ‘A’, jika tidak tampilkan ‘B’.
3. Buat program menggunakan while loop untuk menampilkan barisan bilangan

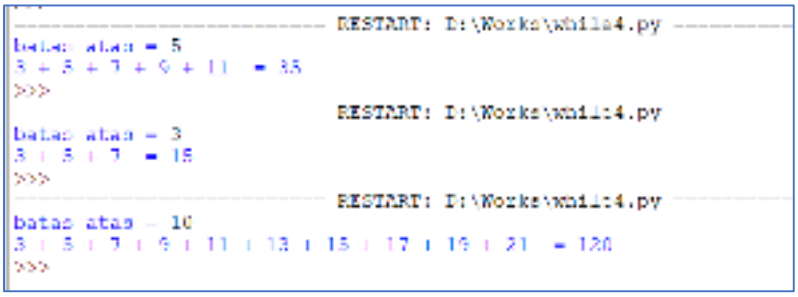
   7 9 11 13 … (2n+5)

4. Program berikut akan mengalikan setiap bilangan yang dimasukkan oleh pengguna, sampai masukkan yang diberikan bernilai 0. Temukan kesalahan pada program berikut, kemudian perbaiki.

```
Kali=1
Bil=int (input(‘bilangan pertama = ‘))
while Bil!=0:
   Bil=int(input(‘Bilangan berikutnya, 0 jika
ingin berhenti) =‘))
```

```
        Kali=Kali*Bil
    print(Kali)
```

5. Modifikasi program pada soal 6, sehingga tampilan outputnya menjadi deret berikut
   7 + 9 + 11 + 13 + …+ (2n+5)
6. Buat program menggunakan while loop untuk menjumlahkan bilangan dari 20 sampai 50
7. Modifikasi program pada soal 6, sehingga batas bawah dan batas atas bilangan yang dijumlahkan ditentukan oleh pengguna.
8. Buat program menggunakan while loop sedemikian hingga tampilan output seperti pada gambar berikut.



9. Tulis dalam struktur for. Untuk nilai k mulai dari 2 sampai 6, dengan nilai k naik dengan penambahan satu, tampilkan nilai k^2.
10. Tuliskan dalam struktur for, untuk x mulai dari 10 sampai 0 dengan penuruan dua-dua, tampilkan nilai x.
11. Ubahlah program berikut menjadi dalam struktur for loop

```
x=8
while (x>0):
   if (x < 5 and x!= 3):
         print(x)
   x=x-1
```

12. Ubahlah program berikut menjadi dalam struktur for loop

```
S=0; x=1
while (x<=5):
      S=S+x
      x=x+3
print(x,S)
```

13. Tentukan output dari program berikut

```
a=10
b=1
for i in range (1,a,2):
   b*=2
print(a,b)
```

14. Tentukan output dari program berikut

```
for i in range(4):
    for k in range(i,0,-1):
        print(i*k,end=" ")
    print()
```

15. Buatlah program menggunakan pengulangan for yang dapat menampilkan keluaran sebagai berikut : (banyak baris merupakan inputan dari pengguna)

```
1
2   3
4   5   6
```

16. Modifikasi program pada Gambar 6.4, sedemikian hingga program akan meminta pengguna untuk memasukkan nilai luas dari suatu persegi panjang jika diketahui ukuran panjang dan lebarnya. Jika jawaban benar maka akan ditampilkan pesan "Selamat, jawaban anda benar", jika tidak pengguna akan diminta menjawab kembali maksimal tiga kali. Buat satu function untuk menghitung luas persegi panjang.
17. Buat program menggunakan for loop untuk menampilkan 10 bilangan bulat antara 20 sampai 50 secara acak.
    (petunjuk: gunakan fungsi *randint*, pada modul *random*).
18. Buat program menggunakan for loop untuk menampilkan semua factor dari suatu bilangan. Pengguna akan memasukkan suatu bilangan bulat positif, kemudian program akan menampilkan semua factor dari bilangan tersebut. Program akan diulang sampai pengguna menyatakan selesai.
19. Buat program menggunakan for loop, untuk menampilkan tabel perkalian bilangan bulat.
20. Buat program untuk menghitung nilai dari

$$\sum_{1}^{n} 2n + 3$$

Gunakan n sebagai inputan dari pengguna.

21. Buat program dimana pengguna akan memasukkan suatu bilangan bulat positif, kemudian program akan menentukan apakah bilangan tersebut bilangan prima atau bukan. Program akan diulang sampai pengguna menyatakan selesai. (catatan: bilangan prima adalah bilangan yang hanya memiliki factor 1 dan dirinya sendiri)
22. Buat program untuk menghitung penjumlahan dua bilangan bulat. Bilangan yang dijumlahkan merupakan bilangan acak antara 1 dan 100. Program akan menampilkan pertanyaan hasil penjumlahan kedua bilangan kepada pengguna. Pengguna menjawab pertanyaan. Jika jawaban pengguna salah, program akan meminta pengguna untuk menjawab kembali sampai jawaban benar. Jika jawaban benar, program akan bertanya apakah pengguna mau mencoba menjawab soal yang lain atau tidak. Jika jawaban ya akan menampilkan soal yang lain, jika tidak akan berhenti. Gunakan function untuk menghitung penjumlahan dengan parameter dua bilangan bulat, dan nilai balikan hasil penjumlahan argument-argumen yang diterimanya.
23. Buatlah program menggunakan for loop. Jika pengguna memasukkan satu bilangan bulat, maka program akan memanggil suatu fungsi untuk menampilkan gambar seperti di bawah ini. Ukuran gambar akan tergantung pada nilai inputan.

BAB 7 STRING

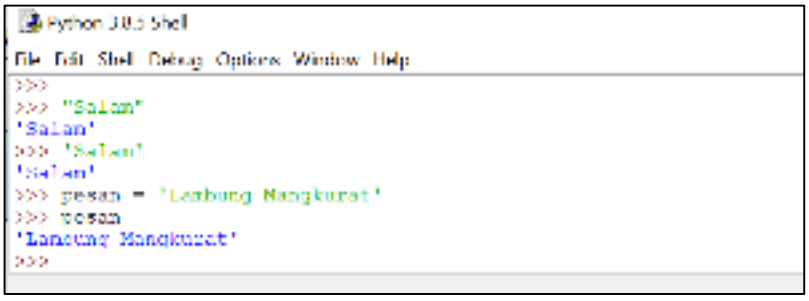
A. Tujuan Pembelajaran

Tujuan yang ingin dicapai dalam pembelajaran ini adalah

1. Diberikan suatu program sederhana terkait penggunaan string, mahasiswa dapat menentukan output program tersebut dengan benar.
2. Diberikan suatu masalah, mahasiswa dapat menghasilkan program untuk menyelesaikan masalah tersebut dengan melibatkan penggunaan string dalam Python dengan jujur dan kreatif.

B. Uraian Materi

Seperti telah diuraikan pada Bab 2, selain bilangan tipe nilai yang juga banyak digunakan dalam pemrograman adalah string. Berbeda dengan bilangan yang merupakan suatu nilai, suatu string merupakan rangkaian karakter. Variabel bertipe string digunakan untuk menyimpan teks, atau rangkaian karakter. Karakter dapat berupa huruf besar atau kecil, angka, spasi, tanda baca, dn symbol-simbol lainnya yang ada pada keyboard. Jika sebagai bilangan 26 merupakan satu nilai, yaitu duapuluh enam. Sebagai string '26', merupakan rangkaian karakter yang terdiri dari angka 2 dan 6. Sejauh ini telah didiskusikan, bahwa apapun karakter yang ada di dalam tanda petik merupakan string dalam Python. Penulisan string sangat fleksibel, kita dapat menulisnya dalam satu tanda petik (single quotes, contoh: 'Salam') atau dua tanda petik (double quotes, contoh: "Salam"). Gambar 7.1 memperlihatkan beberapa contoh pernyataan terkait string.



Gambar 7. 1 Contoh penggunaan string

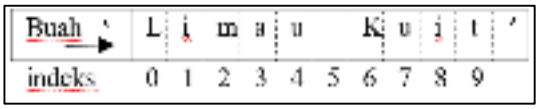
Pada Gambar 7.1, Pernyataan pesan = 'Lambung Mangkurat' merupakan penugasan untuk mengisikan nilai ‘Lambung Mangkurat’ ke variabel pesan. String merupakan rangkaian dari karakter, banyaknya karakter pada suatu string menunjukkan panjang string tersebut. Untuk mengetahui panjang suatu string dapat digunakan fungsi *len*. Perhatikan contoh berikut.

```
>>> Buah == ‘Limau Kuit’
>>> len(Buah)
10
```

Pemanggilan fungsi len dengan argument Buah, akan menghasilkan balikan 10. Hal ini terjadi karena string pada variabel Buah terdiri dari 10 karakter (termasuk spasi).

7.1. Indeks pada String

Masing-masing karakter pada suatu string memiliki nomor indeks. Indeks suatu karakter pada string menunjukkan posisi karakter tersebut di dalam string. Indeks karakter pada string umumnya bilangan bulat positif dan dimulai dari 0. Gambar 7.2 mengilustrasikan system pengindeksan karakter pada string.



Gambar 7. 2 Ilustrasi system indeks pada string

Pada Gambar 7.2, variabel Buah memiliki nilai string ‘Limau kuit’. Indeks dari karakter ‘L’ adalah 0, dan ditulis dengan Buah[0]. Sehingga Buah[1] adalah karakter ‘i’, Buah[2] karakter ‘m’, dan seterusnya. Bila ingin menggunakan system indeks dengan bilangan bulat negative, maka pengindeks-an dimulai dari karakter terakhir. Sebagai contoh, pada variabel Buah, Buah[-1] adalah karakter ‘t’. Indeks dapat juga dinyatakan dengan suatu ekspresi yang memuat variabel, seperti contoh berikut.

```
>>> i = 2
>>> Buah == ‘Limau Kuit’
>>> Buah[i+4]
'K'
```

Karena variabel i bernilai 2, maka Buah[i+4] adalah karakter pada indeks 6. Gambar 7.3 memperlihatkan cara akses karakter pada string menggunakan pengulangan while dan for. Kedua struktur bertujuan untuk menampilkan setiap karakter yang ada pada string dalam variabel Buah.

```
Buah='Limau Kuit'
i=0
while i<len(Buah):
    print (Buah[i])
    i+=1
print()
for huruf in Buah:
    print(huruf)
```

Gambar 7. 3 Contoh akses karakter pada string dengan pengulangan

Bagian dari string disebut *slice*. Misalnya pada variabel Buah, 'mau' merupakan slice dari nilai variabel Buah, yaitu indeks 2, 3, dan 4. Maka dapat ditulis Buah[2:5], yang artinya karakter indeks ke 2 sampai kurang dari 5 pada nilai variabel Buah. Cobalah pernyataan di bawah ini, apa hasilnya?

```
>>> Buah[:3]
>>> Buah[4:]
```

7.2. Bekerja dengan String

Penggabungan beberapa string dapat dilakukan dengan menggunakan operasi penambahan, yang disebut *concatenating string*. Gambar 7.4 memperlihatkan contoh pernyataan *concatenating string*. String pada teks1 di konkatenasi dengan string pada teks2.

```
>>>
>>> teks1 = 'ULM'
>>> teks2 = 'Banjarmasin'
>>> teks1 + ' ' + teks2
'ULM Banjarmasin'
```

Gambar 7. 4 Konkatenasi string

Selain operator + untuk string, kita juga dapat menggunakan operator * pada string dan operator-operator lainnya. Salinlah pernyataan-pernyataan berikut pada shell Python, dan amati hasilnya.

```
>>> 3 * 'Salam '
>>> 'ULM ' * 2
>>> 20 * '-'
>>> 'r' in Buah
>>> 'z' not in Buah
```

Operator relasional juga dapat digunakan pada string. Membandingkan apakah dua string sama atau tidak, dapat dilakukan dengan ekspresi relasional seperti contoh berikut.

```
if Buah == 'Kasturi':
        print('Kalimantan Selatan')
```

Operator relasional lain dapat digunakan, misalkan untuk mengurutkan kata berdasarkan alphabet. Yang perlu diperhatikan adalah bahwa Python akan memunculkan huruf capital sebelum huruf keci, seperti urutan berikut:

```
Pisang, Belimbing, apel
```

Beberapa method dapat digunakan untuk memanipulasi string, diantaranya yang banyak digunakan adalah title(), upper(), dan lower(). Method merupakan suatu aksi yang dapat dilakukan Python terhadap suatu data. Salin dan lengkapi pernyataan-pernyataan berikut pada shell Python, dan amati hasilnya.

```
>>> teks2.upper()
>>> teks2.lower()
>>> teks2.title()
```

Masih banyak method-method lain di dalam Python, anda dapat bereksplorasi untuk mempelajarinya lebih dalam.

7.3. Kesimpulan

1. String merupakan tipe nilai yang merupakan rangkaian karakter.
2. Ukuran atau panjang suatu string dapat diketahui dengan memanggil fungsi len().
3. Untuk mengakses karakter pada string digunakan Indeks.

4. Setiap karakter pada string memiliki indeks yang menunjukkan posisinya di dalam string.

C. Latihan 7

Kerjakan pada shell Python anda.

1. Salin dan jalankan pernyataan berikut

```
>>> T1 = 'dar'
>>> T2 = 'der'
>>> T3 = 'dor'
```

   Buat pernyataan dalam Python, gunakan T1, T2, dan T3 serta operator untuk menampilkan hal berikut
   a. 'dar der dor'
   b. 'dar dar dar dar dar dar dar '
   c. 'dar der der dor dor dor dor'
   d. 'dar dor dar dor dar dor dar dor '
   e. 'derderdor derderdor derderdor derderdor derderdor '
2. Simpanlah nama orang dalam satu variable, kemudian tampilkan ke layar pesan untuk orang tersebut. Contoh hasil: 'Selamat Budi anda lulus'
3. Simpanlah nama orang dalam satu variable, kemudian tampilkan nama tersebut dengan huruf besar semua, huruf kecil semua, dan huruf besar pada huruf awal saja.
4. Untuk string S = 'Bahasa Daerah',
   a. Tuliskan instruksi untuk menampilkan empat karakter pertama
   b. Tuliskan instruksi untuk menampilkan indeks lokasi dari ‘h’ yang pertama
5. Cari kata-kata mutiara dari orang terkenal yang kamu kagumi. Simpan nama orang terkenal dalam variable. Buat pesan pada variable yang lain, kemudian tampilkan nama dan kata-kata tersebut dengan format berikut:
   Albert Einstein berkata, “Seseorang yang tidak pernah berbuat salah tidak pernah mencoba hal yang baru”.
6. Buat program yang memiliki satu fungsi untuk menampilkan string yang diinputkan oleh pengguna dalam huruf kapital semua atau huruf kecil semua sesuai permintaan dari pengguna.

7. Buat program yang memiliki satu fungsi untuk menampilkan dua string yang diinputkan oleh pengguna secara berurutan sesuai alphabet.
8. Buat program yang memiliki suatu fungsi untuk menampilkan inputan string dari pengguna secara terbalik (mulai dari karakter terakhir). Misalkan jika input dari pengguna ‘masuk’ maka akan ditampilkan ‘kusam’.

Kunci Jawaban Soal Pilihan

Latihan 1.

No.1: Alternatif algoritma untuk menghitung luas daerah lingkaran

Algoritma LuasLingk
Input (r)
Luas = 3.14 × r × r
Tampilkan Luas

No.5: Alternatif algoritma menentukan bilangan genap atau ganjil

Algoritma genapGanjil
Input (bil)
A = bil modulo 2
If (A==0) then
    bilangan genap
Else
    bilangan ganjil

Latihan 2.

No 1.c: Ekspresi aritmatik pada Python

```
(3*3-pow(64,1/3))/(5+pow(9,1/2))
```

No 1.d: `435 % 74`

No 2. a: `2 + 9 >= 9`

No 3.c: `z = 2*x + pow(y,3) - 6`

Latihan 3:

No 8:
```
# Alternatif program pengurangan dua pecahan
from fractions import Fraction
a=Fraction(input())
b=Fraction(input())
c=a-b
print(a, " - ", b, " = ", c)
```

No 12:
```
# Alternatif program pembagian dengan 7
a=int(input())
b= a//7
```

```
c= a%7
print(b, c)
```

Latihan 4:

No 1:
```
if waktu == 30:
      Hasil = total*1.5
```

No 5:
```
if penjualan <= 50000:
      Bonus = 0.10
else:
      Bonus = 0.20
```

No 8:
```
# Alternatif program
print('kamu')
b=input('Bahasa Inggrisnya adalah: ')
if b=='you' or b=='You' or b=='YOU':
    print('Selamat anda benar')
else:
    print('Anda salah, belajar lagi ya')
```

Latihan 5:

No 17:
```
# definisi fungsi pembagian dengan 3
def bagi3(a):
      return a/3
```

No 20:
```
tampilkan(a, b, c)
```

No 22:
```
# Alternatif program menghitung Z(x)
import math
def z(x):
    if x>0:
        return math.sqrt(x)+x+1
    elif x<0:
        return pow(x,3)-x+1
    else:
        return 0
#main
x= int(input('x='))
print('z(',x,')= ',z(x))
```

Latihan 6:
No 16: # Program luas persegi panjang

```
import random
def Luas(p,l):
    return p*l

#main
p=random.randint(1,15)
l=random.randint(1,15)
print("Suatu persegi panjang diketahui:")
print("panjang = ", p,'cm dan lebar = ',l,'cm')
print("Berapakah cm persegi luas persegi panjang tersebut?")
for i in range(3):
   L=int(input('Luas= '))
   if L==Luas(p,l):
       print("Selamat jawaban anda benar")
       print('Luas=', p, 'x', l, '=',Luas(p,l),'cm\u00b2')
         break
   else:
        if i!= 2:
            print("Jawaban anda salah, coba lagi")
        else:
        print("Jawaban anda salah, Semangat belajar lagi ya!")
```

No 17: # Program menampilkan 10 bilangan bulat antara 20 sampai 50 secara acak.

```
    i=1
    while i<11:
        print(random.randint(20,50))
         i+=1
```

DAFTAR PUSTAKA

Dierbach, C. (2013). Introduction to Computer Science Using Python: A Computational Problem-Solving Focus. John Wiley & Sons. USA

Perkovic, Lj. (2015). Introduction to Computing Using Python: An Application Development Focus. 2nd ed. John Wiley & Sons Inc. USA

Matthes, E. (2019). Python Crash Course: A Hands-on, Project-Based Introduction to Programming. 2nd ed. No Starch Press Inc. San Francisco.

Downey, A. (2015). Think Python. How to Think Like a Computer Scientist. 2nd ed. Green Tea Press. Needham, Massachusetts.

https://www.w3schools.com/python/

INDEKS

PEMROGRAMAN DASAR

Menggunakan Python

Buku ajar ini merupakan bagian dari proses belajar mengajar untuk mata kuliah Pemrograman Dasar. Buku ini berisi sepuluh bab. Masing-masing diawali dengan pendahuluan dan uraian tentang capaian pembelajaran akhir, diikuti dengan uraian materi, rangkuman dan soal latihan. Alternatif penyelesaian beberapa soal latihan dapat dilihat pada bagian akhir buku ini. Buku ajar ini diharapkan dapat membantu mahasiswa memahami lebih dalam konsep-konsep pemrograman menggunakan python dan aplikasinya.

Penerbit Deepublish (CV BUDI UTAMA)
Jl. Kaliurang Km 9,3 Yogyakarta 55581
Telp/Fax : (0274) 4533427
Anggota IKAPI (076/DIY/2012)
cs@deepublish.co.id
Penerbit Deepublish
@penerbitbuku_deepublish
www.penerbitdeepublish.com

Kategori : Pemrograman

ISBN 978-623-02-3507-8