ISBN 978-634-256-611-5 (PDF)

DASAR-DASAR PEMROGRAMAN PYTHON:

PENDEKATAN PRAKTIS UNTUK PEMULA

PENULIS:

Riki Winanjaya, M.Kom.
Bunga Sabila
P.P.P.A.N.W.Fikrul Ilmi R.H.Zer, M.Kom.
Widodo Saputra, M.Kom.
Masri Wahyuni, M.Kom.

DASAR-DASAR PEMROGRAMAN PYTHON: PENDEKATAN PRAKTIS UNTUK PEMULA

Penulis:
Riki Winanjaya, M.Kom.
Bunga Sabila
P.P.P.A.N.W. Fikrul Ilmi R.H.Zer, M.Kom.
Widodo Saputra, M.Kom.
Masri Wahyuni, M.Kom.

GET PRESS INDONESIA

DASAR-DASAR PEMROGRAMAN PYTHON: PENDEKATAN PRAKTIS UNTUK PEMULA

Penulis :
Riki Winanjaya, M.Kom.
Bunga Sabila P.P.P.A.N.W.
Fikrul Ilmi R.H.Zer, M.Kom.
Widodo Saputra, M.Kom.
Masri Wahyuni, M.Kom.

ISBN : 978-634-255-611-5

Editor : Hendra Nusa Pratama, M.Kom.
Desain Sampul dan Tata Letak : Tri Putri Wahyuni, S.Pd

PENERBIT GET PRESS INDONESIA
Anggota IKAPI No. 033/SBA/2022
Jl. Palarik RT 01 RW 06 Kelurahan Air Pacah
Kecamatan Koto Tangah Padang Sumatera Barat
website: www.getpress.co.id
email: adm.getpress@gmail.com

Cetakan Pertama, Juli, 2026

KATA PENGANTAR

Puji syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa, karena berkat rahmat dan karunia-Nya buku berjudul "Dasar-Dasar Pemrograman Python: Pendekatan Praktis untuk Pemula" ini dapat diselesaikan dengan baik. Buku ini disusun untuk membantu mahasiswa, khususnya mereka yang baru memulai perjalanan dalam dunia pemrograman, agar dapat memahami konsep dasar Python secara terstruktur, sederhana, dan mudah diterapkan.

Materi dalam buku ini merangkum pembelajaran selama satu semester perkuliahan, mulai dari pengenalan dasar Python, variabel dan tipe data, operator, struktur kontrol, fungsi, hingga pengelolaan berbagai jenis collection dan studi kasus praktis. Setiap bab disertai dengan contoh program, latihan praktikum, dan latihan teori agar pembaca tidak hanya memahami konsep secara teoritis, tetapi juga mampu menerapkannya dalam penyelesaian masalah nyata.

Kami berharap buku ini dapat menjadi referensi yang bermanfaat bagi siswa, mahasiswa, dosen, maupun pembaca umum yang ingin mempelajari Python dari dasar dengan pendekatan yang sistematis dan langsung dapat dipraktikkan. Kami menyadari bahwa buku ini masih memiliki kekurangan. Oleh karena itu, kritik dan saran yang membangun sangat kami harapkan demi penyempurnaan di masa mendatang.

Akhir kata, semoga buku ini dapat memberikan manfaat dan menjadi langkah awal yang baik bagi pembaca untuk memahami dunia pemrograman lebih dalam lagi.

Pematangsiantar, Februari 2026
Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENGENALAN PYTHON DAN VARIABEL DASAR

1.1 Pendahuluan

Python merupakan salah satu bahasa pemrograman modern yang banyak digunakan dalam berbagai bidang, mulai dari pengembangan web, analisis data, hingga kecerdasan buatan. Keunggulan Python terletak pada sintaksnya yang sederhana sehingga mudah dipahami, namun tetap memiliki kemampuan yang sangat luas. Oleh karena itu, Python sering dipilih sebagai bahasa pertama bagi pemula yang ingin mempelajari pemrograman.

1.2 Sejarah Singkat dan Keunggulan Python

Python pertama kali diperkenalkan oleh Guido van Rossum pada tahun 1990. Hingga kini, bahasa ini terus berkembang dan menjadi salah satu bahasa pemrograman dengan komunitas terbesar di dunia.

Beberapa keunggulan Python antara lain:

a. Sintaks sederhana: mendekati bahasa manusia, mudah dibaca.
b. Open-source: dapat digunakan secara gratis dan terus dikembangkan oleh komunitas.
c. Multi-platform: dapat dijalankan pada berbagai sistem operasi (Windows, macOS, Linux). d. Kaya pustaka: ribuan library siap digunakan untuk beragam kebutuhan, seperti machine learning, visualisasi data, maupun pengembangan aplikasi.

1.3 Menulis Program Pertama dengan Python

Setiap pembelajaran bahasa pemrograman biasanya diawali dengan menulis program sederhana untuk menampilkan teks ke layar. Tradisi ini dikenal dengan nama "Hello World", yang pertama kali dipopulerkan oleh Brian Kernighan pada tahun 1972 dalam dokumentasi bahasa pemrograman C (Surbakti et al., 2024).

Contoh Program

Input:

```
1   print("Selamat data di mata kuliah Python!")
```

Gambar 1.1. Menampilkan Teks

Output:

```
Selamat data di mata kuliah Python!
```

Gambar 1.2. Output Teks

Fungsi print() digunakan untuk menampilkan teks. Teks harus diletakkan di dalam tanda kutip.

1.4 Interaksi dengan Pengguna

Program komputer tidak hanya menampilkan keluaran, tetapi juga dapat menerima masukan dari pengguna. Dalam Python, fungsi input() digunakan untuk meminta data dari pengguna.

Contoh Program

Input:

```
1   nama = input("Masukkan nama Anda: ")
2   umur =  int(input("Masukkan umur Anda: "))
3   print(f"Halo {nama}! umur Anda {umur} tahun")
4
```

Gambar 1.3. Input Nama

Output:

```
Masukkan nama Anda: Bunga Sabila
Masukkan umur Anda: 20
Halo Bunga Sabila! umur Anda 20 tahun
```

Gambar 1.4. Output dari Input Nama

Penjelasan:

a. input() meminta data dari pengguna.
b. Data yang diperoleh bertipe string.
c. {nama} dan {umur} di dalam f-string digunakan untuk menampilkan nilai variabel.

Fungsi int() digunakan untuk mengubah masukan dari string menjadi integer.

1.5 Variabel dan Tipe Data Dasar

Variabel digunakan untuk menyimpan data. Python menentukan tipe data variabel secara otomatis sesuai dengan nilai yang diberikan (Rahman et al., 2023).

Contoh Program

Input:

```
1   # Variabel dasar
2   nama = "Bunga"        # String
3   umur = 20             # Integer
4   tinggi = 170.5        # Float
5   is_mahasiswa = True   # Boolean
6
7   print("Nama:", nama)
8   print("Umur:", umur)
9   print("Tinggi:", tinggi)
10  print("Mahasiswa:", is_mahasiswa)
```

Gambar 1.5. Variabel Dasar

Output:

```
Nama: Bunga
Umur: 20
Tinggi: 170.5
Mahasiswa: True
```

Gambar 1.6. Output dari Variabel Dasar

Tipe data dasar dalam Python meliputi:

a. Integer (int): bilangan bulat, contoh 20.
b. Float (float): bilangan desimal, contoh 170.5
c. String (str): teks, contoh “Bunga”.
d. Boolean (bool): logika benar/salah, contoh True.

1.6 Studi Kasus

Pada bagian ini akan diberikan beberapa studi kasus sederhana untuk memperkuat pemahaman mengenai penggunaan perintah print(), variabel, tipe data, serta operator matematika di Python. Studi kasus ini membantu mahasiswa melihat bagaimana konsep dasar yang sudah dipelajari dapat langsung diterapkan dalam program sederhana.

a. Menampilkan Judul Program
   Program berikut menampilkan judul dengan garis pemisah menggunakan karakter = yang diulang sebanyak 50 kali.
   Input:

```
1   print("=" * 50)
2   print("BELAJAR DASAR-DASAR PYTHON")
3   print("=" * 50)
```

Gambar 1.7. Menampilkan Judul Program

Output:

```
==================================================
BELAJAR DASAR-DASAR PYTHON
==================================================
```

Gambar 1.8. Output Menampilkan Judul Program

Penjelasan:

1. Ekspresi "=" * 50 menghasilkan string yang berisi tanda = sebanyak 50 kali.
2. Baris kedua menampilkan teks utama sebagai judul.
3. Baris ketiga kembali mencetak garis pemisah, sehingga tampilan terlihat rapi.

b. Menampilkan Judul Sub-Bagian

Program berikut menampilkan judul sub-bagian dengan garis pemisah menggunakan karakter -.

Input:

```
1   print("\n1. VARIABEL DAN TIPE DATA")
2   print("-" * 30)
```

Gambar 1.9. Menampilkan Judul Sub-Bagian

Output:

```
1. VARIABEL DAN TIPE DATA
------------------------------
```

Gambar 1.10. Output Judul Sub-Bagian

Penjelasan:

1. Karakter khusus \n berfungsi untuk membuat baris baru.
2. Baris pertama menampilkan teks judul sub-bagian.
3. Baris kedua mencetak tanda - sebanyak 30 kali sebagai garis pemisah.

c. Menampilkan Variabel String dan Tipenya

Program ini mendemonstrasikan bagaimana mendefinisikan variabel bertipe string dan menampilkan tipe datanya dengan fungsi type().

Input:

```
nama = "Bunga Sabila"
print("Nama:", nama)
print("Tipe data:", type(nama))
```

Gambar 1.11. Menampilkan Variabel String dan Tipenya

Output:

```
Nama: Bunga Sabila
Tipe data: <class 'str'>
```

Gambar 1.12. Output Variabel String dan Tipenya

Penjelasan:

1. Variabel nama menyimpan teks “Bunga Sabila”.
2. Fungsi print() menampilkan nilai variabel ke layar.
3. Fungsi type() digunakan untuk mengetahui tipe data dari variabel.

d. Operator Matematika

Program ini menggunakan berbagai operator matematika pada dua buah variabel, a dan b.

Input:

```
print("\n2. OPERATOR MATEMATIKA")
print("-" * 30)

a = 10
b = 3

print("a =",a, ", b =", b)
print("Penjumlahan (a + b):", a + b)
print("Pengurangan (a - b):", a - b)
print("Perkalian  (a * b):", a * b)
print("Pembagian  (a / b):", a / b)
print("Pangkat  (a ** b):", a ** b)
print("Sisa Bagi/Modulus  (a % b):", a % b)
print("Pembagian Bulat  (a // b):", a // b)
```

Gambar 1.13. Operator Matematika

Output:

```
2. OPERATOR MATEMATIKA
------------------------------
a = 10 , b = 3
Penjumlahan (a + b): 13
Pengurangan (a - b): 7
Perkalian  (a * b): 30
Pembagian  (a / b): 3.3333333333333335
Pangkat  (a ** b): 1000
Sisa Bagi/Modulus  (a % b): 1
Pembagian Bulat  (a // b): 3
```

Gambar 1. 14. Output Operator Matematika

Penjelasan:

1. Variabel a bernilai 10, b bernilai 3.
2. Operator + untuk penjumlahan, - untuk pengurangan, dan * untuk perkalian.
3. Operator / menghasilkan pembagian dengan hasil desimal.
4. Operator ** digunakan untuk perpangkatan.
5. Operator % menghasilkan sisa bagi.
6. Operator // menghasilkan hasil pembagian bulat.

Dengan keempat studi kasus ini, mahasiswa dapat melihat langsung bagaimana Python digunakan untuk:

a. Membuat tampilan terformat dengan karakter khusus.
b. Menampilkan judul bagian secara rapi.
c. Mendefinisikan variabel string dan mengetahui tipenya.
d. Melakukan operasi matematika dasar dengan berbagai operator.

1.7 RANGKUMAN

1. Python adalah bahasa pemrograman modern yang mudah dipahami karena sintaksnya sederhana namun tetap kuat. Bahasa ini open-source, dapat dijalankan di berbagai sistem operasi, dan memiliki banyak pustaka siap pakai.

2. Python diciptakan oleh Guido van Rossum pada tahun 1991 dan berkembang pesat hingga kini, menjadi salah satu bahasa dengan komunitas terbesar di dunia.
3. Program pertama yang dipelajari biasanya adalah Hello World menggunakan fungsi print().
4. Python memungkinkan interaksi dengan pengguna melalui fungsi input(). Data hasil input default bertipe string, namun dapat dikonversi menjadi tipe lain, misalnya integer.
5. Variabel digunakan untuk menyimpan data, sedangkan tipe data dasar Python mencakup integer, float, string, dan boolean.

BAB 2
VARIABEL, KONSTANTA DAN TIPE DATA DALAM PYTHON

2.1 Pendahuluan

Setelah memahami dasar-dasar Python pada bab sebelumnya, kini kita akan melangkah lebih jauh dengan mempelajari variabel, konstanta, dan tipe data. Salah satu langkah awal yang harus dikuasai dalam pemrograman adalah bagaimana menyimpan data. Dalam Python, kita bisa menggunakan variabel untuk menampung data yang dapat berubah-ubah, serta konstanta untuk menyimpan nilai tetap yang seharusnya tidak diubah. Selain itu, Python juga memiliki berbagai tipe data yang memungkinkan kita mengelompokkan informasi sesuai dengan kebutuhan, mulai dari angka, teks, hingga nilai logika. Pemahaman tentang variabel, konstanta, dan tipe data ini menjadi fondasi penting sebelum mempelajari konsep pemrograman yang lebih lanjut.

2.1 Variabel dan Konstanta

Variabel adalah nama yang digunakan untuk menyimpan sebuah nilai agar bisa dipanggil kembali di dalam program. Python bersifat dinamis, artinya kita tidak perlu mendeklarasikan tipe data sejak awal, cukup dengan memberikan nilai, Python otomatis mengenali tipenya.

Contoh:

```
1   nama = "Bunga"
2   usia = 20
```

Gambar 2.1. Variabel

a. Nama berisi teks (string)
b. Usia berisi angka bulat (integer)

Aturan penamaan variabel dalam Python:

1. Nama variabel diawali huruf atau garis bawah _, tidak boleh diawali angka.
2. Karakter selanjutnya boleh huruf, angka, atau garis bawah.
3. Python bersifat case-sensitive → data dan Data dianggap berbeda.
4. Tidak boleh menggunakan kata kunci Python (if, for, dll.).
5. Gunakan nama deskriptif agar mudah dipahami.

Sementara itu, Konstanta merupakan nilai yang seharusnya tidak diubah selama program berjalan. Walaupun Python tidak memiliki aturan khusus untuk konstanta, secara konvensi penulisannya menggunakan huruf kapital.
Contoh:

```
1    PHI = 3.14159
```

Gambar 2.2. Angka Phi

2.3 Aturan Penamaan Tabel

Dalam Python, ada beberapa aturan yang wajib dipatuhi saat membuat nama variabel:

a. Nama variabel harus diawali huruf atau garis bawah (_), tidak boleh angka.
b. Python bersifat case-sensitive → data dan Data dianggap berbeda.
c. Tidak boleh menggunakan kata kunci Python (if, for, class, dll.).
d. Sebaiknya gunakan nama yang deskriptif agar mudah dipahami.

2.4 Tipe Data dalam Python

Dalam pemrograman, tipe data adalah jenis nilai yang bisa disimpan di dalam variabel. Python memiliki sistem tipe data yang dinamis, artinya kita tidak perlu mendeklarasikan tipe data secara eksplisit; Python akan menentukannya berdasarkan nilai yang diberikan.

a. Tipe Data Numerik
   Integer (int): bilangan bulat tanpa desimal.

Contoh:

```
umur = 19
tahun = -2025
```

Gambar 2.3. Angka Integer

Float (float): bilangan desimal.
Contoh:

```
ipk = 3.89
tinggi = 165.5
```

Gambar 2.4. Angka Float

b. Tipe Data String
String adalah teks yang ditulis dengan tanda petik tunggal '...' atau ganda "...". Contoh:

```
nama = "Bunga"
kota = 'Pematangsiantar'
```

Gambar 2.5. Tipe Data String

Untuk teks panjang, gunakan triple quotes:
Contoh:

```
alamat = """Jl. Melati No.10
Kelurahan Siantar
Sumatera Utara"""
```

Gambar 2.6. Tipe Data Triple Quotes

c. Tipe Data Boolean
Boolean hanya memiliki dua nilai: True atau False. Tipe data ini sangat penting dalam logika pemrograman.
Contoh:

```
is_student = True
sudah_lulus = False
```

Gambar 2.7. Tipe Data Boolean

2.5 Studi Kasus

Berikut contoh program sederhana yang memanfaatkan variabel, konstanta, dan berbagai tipe data.

Input:

```
# Konstanta
NAMA_KAMPUS = "STIKOM Tunas Bangsa"

# Variabel
nama = "Bunga Sabila"
nim = 23020009
ipk = 3.89
sudah_lulus = False
semester = 4

# Output
print("=== DATA MAHASISWA ===")
print("Nama:", nama)
print("NIM:", nim)
print("IPK:", ipk)
print("Status Lulus:", sudah_lulus)
print("Kampus:", NAMA_KAMPUS)
```

Gambar 2.8. Studi Kasus Data Mahasiswa

Output:

```
=== DATA MAHASISWA ===
Nama: Bunga Sabila
NIM: 23020009
IPK: 3.89
Status Lulus: False
Kampus: STIKOM Tunas Bangsa
```

Gambar 2.9. Output Studi Kasus Data Mahasiswa

2.6 RANGKUMAN

a. Variabel adalah wadah untuk menyimpan data yang nilainya dapat berubah.
b. Konstanta menyimpan nilai tetap, biasanya ditulis dengan huruf kapital.

c. Aturan penamaan variabel: diawali huruf/underscore, case-sensitive, tidak boleh pakai keyword, gunakan nama deskriptif.
d. Tipe data Python: integer, float, string, boolean.
e. Studi kasus memperlihatkan cara menyimpan data mahasiswa dan menampilkannya dengan print().
f. Latihan praktikum berisi pembuatan variabel, perhitungan lingkaran, konversi suhu, panjang string, dan fungsi type().
g. Latihan teori membahas operator perbandingan, operasi aritmatika, formatting string, menghitung diskon, dan penggunaan list.

BAB 3
OPERATOR DALAM PYTHON

3.1 Pendahuluan

Dalam pemrograman, operator adalah simbol khusus yang digunakan untuk melakukan operasi terhadap data. Python menyediakan berbagai macam operator yang dapat membantu kita dalam melakukan perhitungan matematis, membandingkan nilai, menggabungkan kondisi logika, hingga memeriksa apakah sebuah nilai terdapat dalam suatu koleksi. Pemahaman terhadap operator sangat penting karena hampir setiap baris kode dalam pemrograman memanfaatkan operator.

3.2 Operator Aritmatika

Operator aritmatika digunakan untuk operasi matematika dasar seperti penjumlahan, pengurangan, perkalian, pembagian, hingga perpangkata. Operator yang termasuk di dalamnya: +, -, *, /, //, %, **.

Input:

```
a = 10
b = 3

print("Penjumlahan:", a + b)
print("Pengurangan:", a - b)
print("Perkalian:", a * b)
print("Pembagian:", a / b)
print("Sisa Bagi:", a % b)
print("Pembagian Bulat:", a // b)
print("Perpangkatan:", a ** b)
```

Gambar 3.1. Operasi Aritmatika

Output:

```
Penjumlahan: 13
Pengurangan: 7
Perkalian: 30
Pembagian: 3.3333333333333335
Sisa Bagi: 1
Pembagian Bulat: 3
Perpangkatan: 1000
```

Gambar 3.2. Output Operasi Aritmatika

Penjelasan:
Operator aritmatika membantu dalam perhitungan numerik. Contoh di atas memperlihatkan bagaimana Python menangani berbagai operasi matematika.

3.3 Operator Penugasan (Assigment)

Operator penugasan digunakan untuk memberikan nilai pada variabel. Operator ini bisa digabung dengan operasi aritmatika untuk mempersingkat kode. Beberapa contoh: =, +=, -=, *=, /=.

Input:

```
x = 10
x += 5   # x = 15
x -= 3   # x = 12
x *= 2   # x = 24
x /= 4   # x = 6.0
print("Nilai akhir x:", x)
```

Gambar 3.3. Operasi Penugasan (Assignment)

Output:

```
Nilai akhir x: 6.0
```

Gambar 3.4. Output Operasi Penugasan (Assignment)

Penjelasan:
Operator assignment memudahkan kita memperbarui nilai variabel tanpa menulis ulang perhitungan panjang.

3.4 Operator Perbandingan (Relasional)

Operator perbandingan membandingkan dua nilai. Hasilnya berupa True atau False. Contoh: ==, !=, >, <, >=, <=.
Input:

```
x = True
y = False

print("x and y:", x and y)
print("x or y:", x or y)
print("not x:", not x)
```

Gambar 3.5. Operator Perbandingan

Output:

```
x and y: False
x or y: True
not x: False
```

Gambar 3.6. Output Operator Perbandingan

Penjelasan:
Operator ini penting untuk pengambilan keputusan, misalnya dalam percabangan if.

3.5 Operator Logika

Operator logika dipakai untuk menggabungkan dua kondisi boolean. Ada tiga: and, or, not.
Input:

```
x = True
y = False

print("x and y:", x and y)
print("x or y:", x or y)
print("not x:", not x)
```

Gambar 3.7. Operator Logika

Output:

```
x and y: False
x or y: True
not x: False
```

Gambar 3.8. Output Operator Logika

Penjelasan:
Operator and hanya bernilai benar jika kedua kondisi benar, or bernilai benar jika salah satunya benar, dan not membalikkan nilai boolean.

3.6 Operator Identitas

Digunakan untuk mengecek apakah dua variabel merujuk pada objek yang sama.

a. is → True jika objek sama
b. is not → True jika objek berbeda

Input:

```
a = [1, 2, 3]
b = a
c = [1, 2, 3]

print("a is b:", a is b)
print("a is c:", a is c)
print("a == c:", a == c)
```

Gambar 3.9. Operator Identitas

Output:

```
a is b: True
a is c: False
a == c: True
```

Gambar 3.10. Output Operator Identitas

Penjelasan:
Variabel is mengecek apakah variabel menunjuk objek yang sama, sedangkan == hanya membandingkan nilai.

3.7 Operator Keanggotan

Dipakai untuk memeriksa apakah sebuah nilai ada di dalam koleksi (string, list, tuple).

a. in → True jika nilai ditemukan
b. not in → True jika nilai tidak ditemukan

Input:

```
teks = "Belajar Python"

print("Python" in teks)
print("Java" not in teks)
```

Gambar 3.11. Operator Keanggotaan

Output:

```
True
True
```

Gambar 3.12. Output Operator Keanggotaan

Penjelasan:
Operator keanggotaan sering dipakai dalam pencarian data pada struktur koleksi.

3.8 Studi Kasus

Dalam kehidupan nyata, operator tidak hanya dipakai untuk hitung-hitungan sederhana, tapi juga bisa digunakan dalam sistem penilaian. Misalnya, seorang dosen ingin menentukan apakah mahasiswa lulus atau tidak dalam sebuah mata kuliah. Aturannya:

a. Mahasiswa lulus jika nilai ujian akhir lebih besar atau sama dengan 70.
b. Jika nilai di bawah 70, mahasiswa dinyatakan tidak lulus.

c. Selain itu, mahasiswa yang lulus akan mendapatkan status tambahan berupa “memuaskan” jika nilainya di atas 85.

Input:

```
nilai = 88

status = "Lulus" if nilai >= 70 else "Tidak Lulus"
predikat = "Memuaskan" if nilai > 85 else "Cukup"

print("Nilai:", nilai)
print("Status:", status)
print("Predikat:", predikat)
```

Gambar 3.13. Studi Kasus: Sistem Penilaian

Output:

```
Nilai: 88
Status: Lulus
Predikat: Memuaskan
```

Gambar 3.14. Output Studi Kasus: Sistem Penilaian

Penjelasan:
Studi kasus ini menggunakan kombinasi operator perbandingan (>=, >) dan operator ternary untuk menentukan status kelulusan serta predikat mahasiswa. Dengan studi kasus ini, pembaca bisa melihat bagaimana operator Python dipakai dalam skenario nyata yang sangat dekat dengan dunia akademik.

3.9 RANGKUMAN

Operator di Python digunakan untuk melakukan operasi terhadap variabel dan nilai. Ada beberapa jenis operator yang penting dipahami:

a. Aritmatika → penjumlahan, pengurangan, perkalian, pembagian, sisa bagi, pangkat.
b. Perbandingan → membandingkan dua nilai (>, <, ==, !=, >=, <=).
c. Logika → menggabungkan ekspresi boolean (and, or, not).
d. Assignment → memberikan dan memperbarui nilai variabel (=, +=, -=).

e. Membership → mengecek keberadaan elemen dalam sebuah koleksi (in, not in).
f. Identity → membandingkan apakah dua objek merujuk ke memori yang sama (is, is not).
g. Operator bisa digabungkan untuk menyelesaikan studi kasus nyata, misalnya perhitungan total belanja atau sistem kelulusan mahasiswa.

BAB 4
PERINTAH MASUKAN DAN KELUARAN DALAM PYTHON

4.1 Pendahuluan

Dalam pemrograman, interaksi antara komputer dengan pengguna adalah hal yang sangat penting. Komputer tidak bisa bekerja sendirian tanpa adanya data yang diberikan oleh user. Sebaliknya, user juga perlu tahu hasil dari proses yang dilakukan program. Untuk itulah kita menggunakan perintah masukan (input) dan perintah keluaran (output).

4.2 Perintah Input (Masukkan)

Di Python, perintah input() digunakan untuk menerima data dari pengguna. Data yang dimasukkan melalui input() secara default akan terbaca sebagai string (Prayogo, 2024). Jika ingin menggunakan tipe data lain seperti integer atau float, data tersebut perlu diubah menggunakan fungsi konversi, misalnya:

a. int(input()) → mengubah input ke bilangan bulat.
b. float(input()) → mengubah input ke bilangan pecahan.
   Contoh Sederhana:

```
nama = input("Masukkan nama: ")
umur = int(input("Masukkan umur: "))
print("Halo,", nama, "usia kamu", umur, "tahun")
```

4.3 Perintah Output (Keluaran)

Untuk menampilkan data ke layar, Python menyediakan fungsi print(). Fungsi ini fleksibel karena bisa digunakan untuk:

a. Menampilkan teks biasa.
b. Menampilkan nilai variabel.
c. Menggabungkan teks dengan variabel.
d. Melakukan format tampilan agar lebih rapi.

Contoh Sederhana:

```
nama = "Bunga"
print("Halo,", nama)
```

Python juga mendukung string formatting untuk membuat tampilan lebih jelas, seperti menggunakan f-string.

Contoh Sederhana:

```
nama = "Bunga"
umur = 20
print(f"Halo, nama saya {nama}, umur saya
{umur} tahun")
```

4.4 Pentingnya Input dan Output

Tanpa perintah input, program hanya berjalan statis tanpa interaksi. Sedangkan tanpa output, pengguna tidak tahu hasil dari proses yang dilakukan. Oleh karena itu, input dan output adalah pondasi dasar dari setiap program, termasuk aplikasi besar seperti system kasir, aplikasi e-commerce, hingga sistem informasi akademik.

4.5 Studi Kasus

Sebuah aplikasi sederhana ingin dibuat untuk membantu proses pemesanan tiket bioskop. Program ini akan :

a. Meminta pengguna memasukkan nama, jumlah tiket, dan harga per tiket.
b. Menghitung total harga.
c. Menampilkan ringkasan pemesanan ke layar.

Kode Program:

```
nama = input("Masukkan nama pemesan: ")
jumlah = int(input("Masukkan jumlah tiket: "))
harga = float(input("Masukkan harga per tiket: "))
total = jumlah * harga
print("\n=== RINGKASAN PEMESANAN ===")
print(f"Nama Pemesan : {nama}")
print(f"Jumlah Tiket : {jumlah}")
print(f"Total Bayar : Rp {total:,.0f}")
```

Output:

```
Masukkan nama pemesan: Bunga
Masukkan jumlah tiket: 2
Masukkan harga per tiket: 50000

=== RINGKASAN PEMESANAN ===
Nama Pemesan : Bunga
Jumlah Tiket : 2
Total Bayar  : Rp 100,000
```

Gambar 4.1. Studi Kasus: Sistem Penesanan Tiket

Penjelasan:
Studi kasus ini memanfaatkan input untuk mengambil data dari pengguna dan output untuk menampilkan hasil perhitungan dengan format yang rapi. Tanpa perintah input, data pemesanan tidak bisa dimasukkan, dan tanpa output, pengguna tidak bisa melihat ringkasan hasil transaksi.

4.6 RANGKUMAN

Input adalah data yang dimasukkan pengguna untuk diproses oleh program, sedangkan output adalah hasil dari pemrosesan tersebut yang ditampilkan kembali ke layar.

Python menyediakan dua fungsi utama:

a. input() → menerima masukan dari user (default berupa string). print() → menampilkan informasi ke layar.
b. Input bisa dikonversi ke tipe data lain dengan fungsi int() dan float().

Fungsi print() memiliki variasi:

a. Parameter sep → mengatur pemisah antar argumen.
b. Parameter end → mengatur akhir baris output.

Escape character (\n, \t, \\, dll) membantu mengatur tampilan teks. String formatting membuat output lebih rapi: Metode .format() dan f-string (lebih sederhana dan modern). Studi kasus pada bab ini menunjukkan bagaimana input-output digunakan dalam program nyata, misalnya aplikasi pemesanan tiket atau pengisian biodata.

BAB 5
PERCABANGAN DALAM PEMROGRAMAN PYTHON

5.1 Pendahuluan

Percabangan adalah fondasi logika keputusan dalam program. Dengan percabangan, program bisa memilih jalur eksekusi yang tepat berdasarkan kondisi: “Jika … maka … (selain itu …)”. Gagasan sederhananya: ketika suatu pernyataan bernilai benar, jalankan serangkaian perintah; jika tidak, jalankan jalur lain atau lewati. Inti ini memungkinkan program menjadi responsif terhadap input/situasi yang berbeda.

5.2 Konsep Dasar Percabangan

Percabangan adalah mekanisme evaluasi kondisi yang mengontrol alur program. Hasil evaluasi berupa nilai logika (Benar/Salah), dan keputusan eksekusi diambil berdasarkan hasil itu.

Fungsi ttama membuat untuk program dapat memilih jalur eksekusi yang tepat, menciptakan logika yang responsif terhadap input atau situasi yang berbeda-beda.

Contoh Praktis:

Jika nilai > 0, program mencetak "Positif". Jika tidak, program mencetak "Negatif atau Nol" sederhana namun powerful.

5.3 Struktur Dasar Percabangan: If Statement

Statement if adalah pondasi semua percabangan dalam Python. Kondisi dievaluasi sebagai Boolean (True/False).

Contoh Implementasi:

```
nilai = 85if nilai >=
70:    print("Lulus!")
print("Selamat!")
```

Gambar 5.1. Statement If

Ketika nilai 85 >= 70 bernilai True, kedua baris kode dalam blok if akan dieksekusi secara berurutan.

5.4 Studi Kasus

Membuat sistem penilaian mahasiswa dengan berbagai kriteria.

Kode Program:

```
def minta_float(prompt, min_val=0.0, max_val=100.0):
"""Minta input float dengan validasi rentang nilai
[min_val, max_val]."""" while True:
try:
val = float(input(prompt))
if min_val <= val <= max_val:
return val
print(f"Nilai harus antara {min_val}-{max_val}. Coba
lagi.\n")
except ValueError:
print("Input harus berupa angka. Coba lagi.\n")

def      hitung_nilai_akhir(nilai_tugas,      nilai_uts,
nilai_uas):
"""Bobot: Tugas 30%, UTS 30%, UAS 40%."""
return (nilai_tugas * 0.30) + (nilai_uts * 0.30) +
(nilai_uas * 0.40)

def tentukan_grade(nilai_akhir):
"""
Kembalikan (grade, predikat) sesuai ambang:
>=85 A 'Sangat Memuaskan'
>=80 A- 'Sangat Memuaskan'
>=75 B+ 'Memuaskan'
>=70 B 'Memuaskan'
```

```
>=65 B- 'Cukup Memuaskan'
>=60 C+ 'Cukup'
>=55 C 'Cukup'
>=50 D 'Kurang'
else  E 'Sangat Kurang'
"""
if nilai_akhir >= 85:
      return "A", "Sangat Memuaskan"
elif nilai_akhir >= 80:
      return "A-", "Sangat Memuaskan"
elif nilai_akhir >= 75:
      return "B+", "Memuaskan"
elif nilai_akhir >= 70:
      return "B", "Memuaskan"
elif nilai_akhir >= 65:
      return "B-", "Cukup Memuaskan"
elif nilai_akhir >= 60:
      return "C+", "Cukup"
elif nilai_akhir >= 55:
      return "C", "Cukup"
elif nilai_akhir >= 50:
      return "D", "Kurang"
else:
      return "E", "Sangat Kurang"
def status_kelulusan(nilai_akhir, kehadiran):

if nilai_akhir >= 55 and kehadiran >= 75:
status = "LULUS"
if nilai_akhir >= 85 and kehadiran >= 90:
      catatan = "Dianjurkan untuk mengambil mata kuliah
lanjutan"
elif nilai_akhir >= 70:
      catatan = "Performa baik, pertahankan!"
else:
      catatan   =   "Perlu   peningkatan   di   semester
berikutnya"
else:
      status = "TIDAK LULUS"
if nilai_akhir < 55:
      catatan = "Harus mengulang mata kuliah"
else:
      catatan = "Tidak memenuhi syarat kehadiran"
return status, catatan

def rekomendasi_sks(grade):
"""
```

```
Rekomendasi beban SKS & saran singkat:
- A / A-       : 24 SKS, ambil mata kuliah challenging
- B+ / B / B-  : 21 SKS, pertahankan performa
- C+ / C       : 18 SKS, fokus mata kuliah dasar
- lainnya (D/E) : 12 SKS, wajib konsultasi dosen wali
"""

if grade in ("A", "A-"):
return 24, [
"Dapat mengambil 24 SKS",
"Dianjurkan mengambil mata kuliah challenging",
]
elif grade in ("B+", "B", "B-"):
return 21, [
"Dapat mengambil 21 SKS", "Pertahankan performa
akademik",
]
elif grade in ("C+", "C"):
return 18, [
"Maksimal 18 SKS",
"Perlu fokus pada mata kuliah dasar",
]
else:
return 12, [
"Maksimal 12 SKS",
"Wajib konsultasi dengan dosen wali",
]

def sistem_penilaian_mahasiswa():
print("=" * 50)
print("SISTEM PENILAIAN MAHASISWA")
print("=" * 50)

# Input data
nama = input("Masukkan nama mahasiswa: ")
nim = input("Masukkan NIM: ")
nilai_tugas = minta_float("Masukkan nilai Tugas (0-100):
")
nilai_uts  = minta_float("Masukkan nilai UTS (0-100): ")
nilai_uas  = minta_float("Masukkan nilai UAS (0-100): ")
kehadiran  = minta_float("Masukkan persentase kehadiran
(%): ")

# Proses
nilai_akhir = hitung_nilai_akhir(nilai_tugas, nilai_uts,
nilai_uas)
```

```
grade, predikat = tentukan_grade(nilai_akhir)
status,    catatan    =    status_kelulusan(nilai_akhir,
kehadiran)
sks, saran_sks = rekomendasi_sks(grade)
# Output
print("\n" + "-" * 50) print("HASIL PENILAIAN")
print("-" * 50)
print(f"Nama Mahasiswa : {nama}")
print(f"NIM             : {nim}")
print(f"Nilai Akhir    : {nilai_akhir:.2f}")
print(f"Grade          : {grade}")
print(f"Predikat       : {predikat}")
print(f"Kehadiran      : {kehadiran:.0f}%")
print(f"Status         : {status}")
print(f"Catatan        : {catatan}")
print("\nREKOMENDASI:")
for baris in saran_sks:
print(f"- {baris}")
# Jalankan program
if    __name__ == " __main__":
sistem_penilaian_mahasiswa()
```

Output:

```
==============================================
SISTEM PENILAIAN MAHASISWA
==============================================
Masukkan nama mahasiswa: Bunga Sabila
Masukkan NIM: 230020099
Masukkan nilai Tugas (0-100): 78
Masukkan nilai UTS (0-100): 80
Masukkan nilai UAS (0-100): 90
Masukkan persentase kehadiran (%): 100

----------------------------------------------
HASIL PENILAIAN
----------------------------------------------
Nama Mahasiswa : Bunga Sabila
NIM            : 230020099
Nilai Akhir    : 83.40
Grade          : A-
Predikat       : Sangat Memuaskan
Kehadiran      : 100%
Status         : LULUS
Catatan        : Performa baik, pertahankan!

REKOMENDASI:
- Dapat mengambil 24 SKS
- Dianjurkan mengambil mata kuliah challenging
```

Gambar 5.2. Sistem Penilaian Mahasiswa dengan Berbagai Kriteria

Penjelasan:

Program ini mengelola penilaian akhir mahasiswa: menerima input nilai dan kehadiran, menghitung nilai akhir berbobot, menetapkan grade & predikat, menentukan status kelulusan dengan syarat gabungan (nilai & kehadiran), lalu memberi rekomendasi SKS. Konsep percabangan yang tampak: if/elif/else, nested if, dan operator logika (and) untuk multi-kondisi.

5.5 RANGKUMAN

Percabangan adalah mekanisme "jika… maka…" yang membuat program dapat memilih jalur eksekusi berbeda sesuai kondisi. Fungsi Utama:

a. Membantu program mengambil keputusan.
b. Membuat logika program lebih dinamis dan responsif.
c. Menghasilkan keluaran berbeda sesuai situasi.

Konsep Dasar:

a. Percabangan selalu mengevaluasi kondisi ke bentuk Boolean (True/False).
b. Jika kondisi True, blok aksi dijalankan.
c. Jika kondisi False, blok dilewati atau masuk ke jalur lain.

Struktur Dasar:

a. Pondasi percabangan adalah pernyataan if.
b. if dapat berisi satu atau lebih perintah yang dijalankan secara berurutan ketika kondisi benar.

Penerapan Umum:

a. Percabangan sederhana (if).
b. Percabangan dengan alternatif (if–else).
c. Percabangan dengan banyak kondisi (if–elif–else).

BAB 6
PERULANGAN PADA PEMROGRAMAN PYTHON

6.1 Pendahuluan

Dalam pemrograman, perulangan (looping) digunakan untuk mengeksekusi satu atau beberapa pernyataan secara berulang-ulang selama kondisi tertentu terpenuhi. Konsep perulangan sangat membantu ketika suatu proses perlu dilakukan berkali-kali tanpa menulis kode yang sama berulang. Python menyediakan dua jenis struktur perulangan utama, yaitu for dan while. Keduanya berfungsi untuk mengulangi blok perintah, namun digunakan pada situasi yang berbeda.

6.2 Definisi Perulangan (Loop) di Python

Perulangan adalah mekanisme fundamental dalam pemrograman yang memungkinkan kita menjalankan blok kode secara berulang kali secara otomatis. Dengan perulangan, kita bisa menghemat waktu dan membuat kode lebih efisien. Loop dapat dikombinasikan dengan struktur logika if-else untuk membuat alur program yang dinamis. Studi kasus pada bab ini memperlihatkan penerapan konsep perulangan:

a. Simulasi Bank: penggunaan while untuk proses transaksi berulang.
b. Analisis Nilai Siswa: pemakaian for untuk menghitung rata-rata nilai.
c. Game Tebak Angka: kombinasi while dan logika kondisi untuk permainan interaktif.
d. Manajemen Perpustakaan: pengelolaan data dengan while, for, dan manipulasi list.

Pemahaman terhadap perulangan menjadi dasar penting dalam membangun program yang dinamis, efisien, dan fleksibel.

6.3 Jenis-Jenis Perulangan di Python

Python menyediakan dua jenis perulangan utama yang memiliki kegunaan berbeda sesuai kebutuhan pemrograman. Jenis Perulangan terdapat dua, yaitu:

a. Perulangan For Loop
b. Perulangan While Loop

6.4 Perulangan For Loop

Perulangan For Loop merupakan perulangan yang dilakukan berdasarkan elemen dalam urutan seperti list, tuple, string, atau range. Perulangan ini ideal ketika kita tahu jumlah iterasi yang dibutuhkan, misalnya:

a. Iterasi melalui koleksi data
b. Jumlah pengulangan sudah diketahui dan mudah dibaca

6.5 Perulangan While Loop

Perulangan While Loop meurpakan perulangan dimana selama kondisi tertentu masih bernilai True. Cocok untuk situasi di mana jumlah iterasi tidak diketahui sebelumnya:

a. Bergantung pada kondisi boolean
b. Jumlah pengulangan dinamis
c. Fleksibel untuk logika kompleks

6.6 RANGKUMAN

Perulangan (looping) digunakan untuk mengeksekusi blok kode secara berulang selama kondisi tertentu terpenuhi. Struktur perulangan membantu membuat program menjadi lebih efisien dan mengurangi penulisan kode yang berulang.Python memiliki dua jenis perulangan utama, yaitu:

a. for loop → digunakan ketika jumlah pengulangan diketahui.
b. while loop → digunakan ketika pengulangan bergantung pada kondisi logika tertentu.

Fungsi range() sering digunakan pada for loop untuk menentukan batas pengulangan. Dalam perulangan while, penting untuk memperbarui kondisi agar tidak terjadi infinite loop.

BAB 7
OPERASI STRING PEMROGRAMAN PYTHON

7.1 Pendahuluan

Dalam dunia pemrograman, pengolahan data teks menjadi salah satu aspek penting yang hampir selalu muncul dalam setiap aplikasi. Bahasa Python menyediakan tipe data khusus bernama string untuk merepresentasikan dan memanipulasi teks dengan mudah. Melalui string, programmer dapat menyimpan, menampilkan, dan memproses informasi dalam bentuk karakter seperti huruf, angka, maupun simbol.

Pemahaman tentang string tidak hanya penting untuk membuat program sederhana seperti menampilkan pesan atau membaca input pengguna, tetapi juga menjadi fondasi bagi proses yang lebih kompleks seperti analisis teks, pemrosesan data masif, dan pengembangan antarmuka pengguna berbasis teks.

Pada bab ini akan dipelajari berbagai teknik pengolahan string dalam Python, meliputi cara pembuatan, penggabungan, pemotongan, penggunaan karakter khusus (escape characters), hingga penerapan berbagai metode bawaan (built-in methods) yang memudahkan manipulasi teks.

Melalui pembahasan ini, diharapkan pembaca dapat memahami bagaimana teks dikelola di dalam Python serta mampu menerapkannya untuk berbagai kebutuhan praktis dalam pemrograman.

7.2 Pengertian String

String adalah kumpulan karakter yang dikelilingi tanda kutip tunggal ('...'), ganda ("..."), atau tripel ('''...'). Python tidak memiliki tipe data char, jadi string dianggap sebagai daftar karakter. Setiap karakter di-encode dalam Unicode,

memungkinkan penggunaan berbagai simbol dan bahasa dari seluruh dunia.

7.3 Membuat dan Mengakses String

String bersifat immutable, artinya isinya tidak dapat diubah setelah dibuat. Namun, bagian tertentu dari string dapat diakses menggunakan indeks atau slicing.

Contoh:

```
kata = "Python"
print(kata[0])    # huruf pertama
print(kata[-1])   # huruf terakhir
print(kata[0:4]) # potongan string
```

7.4 Menggabungkan dan Mengulang String

Python mendukung operasi aritmetika sederhana pada string. Operator + digunakan untuk menggabungkan string, sedangkan * digunakan untuk mengulang string beberapa kali.

Contoh:

```
a = "Hello"
b = "World"
print(a + " " + b)
print(a * 3)
```

7.5 Karakter Khusus dan Escape Character

Untuk menampilkan karakter tertentu seperti tanda kutip, tab, atau baris baru, digunakan escape character dengan tanda \.

Contoh:

```
print("Baris pertama\nBaris kedua")
print("Nama:\tBunga")
print("Tanda kutip ganda: \"Python\"")
```

7.6 String Formating

Python menyediakan beberapa cara untuk menampilkan teks dengan nilai variabel secara dinamis:

a. Menggunakan koma (,)
b. Menggunakan .format()
c. Menggunakan f-string

Contoh:

```
nama = "Bunga" umur = 20
print("Nama:", nama, ", Umur:", umur)
print("Nama: {} | Umur: {}".format(nama, umur))
print(f"Nama: {nama} | Umur: {umur}")
```

7.7 Fungsi dan Metode String Bawaan

Python memiliki berbagai metode bawaan (built-in methods) untuk manipulasi teks:

Tabel 7.1. Fungsi dan Metode String Bawaan

Metode	Fungsi
.upper()	Mengubah semua huruf menjadi kapital
.lower()	Mengubah semua huruf menjadi huruf kecil
.capitalize()	Mengubah huruf pertama menjadi kapital
.title()	Mengubah huruf awal setiap kata menjadi kapital
.replace(a, b)	Mengganti substring a dengan b
.split()	Memisahkan string menjadi list
.strip()	Menghapus spasi di awal dan akhir string
.find()	Mencari posisi substring
.count()	Menghitung jumlah kemunculan substring

Contoh:

```
teks = "Belajar Python Sangat Menyenangkan"
print(teks.upper())
print(teks.lower())
print(teks.replace("Menyenangkan", "Mudah"))
print(teks.count("a"))
```

7.8 Menghitung Panjang String

Fungsi len() digunakan untuk menghitung jumlah karakter dalam sebuah string, termasuk spasi.

Contoh:

```
kalimat = "Pemrograman Python"
print(len(kalimat))
```

7.9 RANGKUMAN

a. Python memiliki dukungan kuat untuk mengolah teks melalui tipe data string dan berbagai metode bawaan seperti replace(), split(), join(), strip(), dan find().
b. Manipulasi teks memungkinkan pembuatan program seperti text analyzer, formatter, dan validator.
c. Konsep kelas dan objek digunakan untuk membangun sistem modular, seperti TextAnalyzer, StringManipulator, dan TextFormatter.
d. Setiap kelas memiliki tanggung jawab khusus, misalnya analisis kata, pembalikan teks, validasi format data, dan pengukuran panjang rata-rata kata.

BAB 8
OPERASI BILANGAN PEMROGRAMAN PYTHON

8.1 Pendahuluan

Dalam pemrograman, pengolahan nilai numerik merupakan salah satu kemampuan dasar yang perlu dipahami sejak awal. Python mendukung berbagai jenis operasi aritmatika yang memungkinkan pengguna untuk melakukan perhitungan matematis, baik sederhana maupun kompleks.

Bilangan dalam Python dapat direpresentasikan dalam bentuk integer (bilangan bulat) dan float (bilangan pecahan). Untuk memanipulasi bilangan tersebut, Python menyediakan sekumpulan operator yang digunakan untuk melakukan operasi seperti penjumlahan, pengurangan, perkalian, pembagian, dan lain sebagainya.

Pemahaman mengenai operator aritmatika sangat penting karena menjadi dasar dalam membangun logika program, baik dalam aplikasi sederhana (misalnya kalkulator), maupun kompleks seperti sistem simulasi dan perhitungan statistik.

8.2 Operator

Operator adalah simbol yang digunakan untuk memberi perintah agar suatu operasi dilakukan terhadap operand. Dalam konteks operasi bilangan, operator digunakan untuk menjumlahkan, mengurangi, mengalikan, membagi, atau melakukan perhitungan lainnya.

Contoh operator:

```
+, -, *, /, %, //, **
```

8.3 Operand

Operand merupakan nilai atau variabel yang menjadi objek dari suatu operasi. Operand dapat berupa bilangan bulat

(integer), bilangan pecahan (float), maupun variabel yang menyimpan nilai numerik.
Contoh operand:
8, 5, x, jumlah

8.4 Jenis-Jenis Operator Aritmatika dalam Python

Tabel 8.1. Jenis-Jenis Operator Aritmatika

Operator	Nama	Fungsi	Contoh	Hasil
+	Penjumlahan	Menjumlahkan dua nilai	5 + 3	8
-	Pengurangan	Mengurangi nilai	10 - 4	6
*	Perkalian	Mengalikan dua nilai	6 * 7	42
/	Pembagian	Menghasilkan nilai pecahan	15 / 4	3.75
%	Modulus	Mengambil sisa pembagian	17 % 5	2
**	Pangkat	Pemangkatan bilangan	2 ** 3	8
//	Pembagian Bulat	Menghasilkan nilai bulat tanpa desimal	15 // 4	3

8.5 RANGKUMAN

a. Bilangan dalam Python dapat berupa integer (bilangan bulat) dan float (bilangan pecahan).
b. Operator adalah simbol yang digunakan untuk melakukan operasi perhitungan, sedangkan
c. operand adalah nilai yang dikenai operasi tersebut.
d. Python menyediakan beberapa operator aritmatika, antara lain:
    - + penjumlahan
    - - pengurangan
    - * perkalian
    - / pembagian
    - % modulus (sisa pembagian)
    - // pembagian bulat
    - ** pemangkatan
e. Operator aritmatika dapat digunakan pada nilai langsung maupun variabel.

f. Hasil operasi pembagian / selalu menghasilkan tipe data float, sedangkan // menghasilkan nilai integer tanpa pecahan.
g. Pemahaman operasi bilangan menjadi dasar dalam pembuatan program, terutama sebelum mempelajari percabangan, perulangan, dan struktur data.

BAB 9
STRUKTUR DATA DALAM PYTHON

9.1 Pendahuluan

Struktur data merupakan salah satu komponen paling penting dalam dunia pemrograman. Melalui struktur data, program dapat menyimpan, mengatur, dan memanipulasi informasi dengan cara yang efisien dan mudah diakses.

Python menyediakan beragam struktur data bawaan yang bersifat dinamis, fleksibel, dan sangat efisien untuk berbagai kebutuhan pengolahan data.

Pemahaman terhadap struktur data menjadi fondasi utama dalam mengembangkan aplikasi yang tangguh dan efisien mulai dari program sederhana hingga sistem kompleks seperti analisis data, kecerdasan buatan (Artificial Intelligence), dan pengelolaan basis data berskala besar.

9.2 Pengertian Struktur Data

Struktur data adalah cara untuk menyimpan dan mengatur data agar mudah diakses dan dimodifikasi di dalam program. Python memberikan dukungan penuh terhadap struktur data bawaan seperti List, Tuple, Set, dan Dictionary, yang masing-masing memiliki karakteristik dan fungsi yang berbeda.
Fungsi dan Manfaat Struktur Data:

a. Efisiensi Program, pemilihan struktur data yang tepat meningkatkan kecepatan dan performa program.
b. Pengelolaan Data, memudahkan dalam manipulasi serta akses data yang kompleks.
c. Fleksibilitas, memungkinkan pengembang menyesuaikan penyimpanan data dengan kebutuhan aplikasi.

9.3 Jenis-Jenis Struktur Data dalam Python

a. List

List merupakan kumpulan Data Berurutan dan Dapat Diubah. List merupakan struktur data paling fleksibel di Python. Elemen-elemen dalam list bersifat ordered (berurutan) dan mutable (dapat diubah).

Ciri-Ciri List:

1. Setiap elemen memiliki posisi tetap yang diakses melalui indeks (dimulai dari 0).
2. Elemen list dapat diubah, ditambah, atau dihapus.
3. Satu list dapat menyimpan berbagai tipe data (angka, string, boolean, bahkan list lain).

Contoh Program:

```
buah = ["apel", "jeruk", "mangga"]
buah.append("pisang")
print(buah)
# Output: ["apel", "jeruk", "mangga",
"pisang"]
```

b. Tuple

Tuple merupakan kumpulan Data Berurutan dan Tidak Dapat Diubah. Tuple hampir sama dengan list, namun bersifat immutable, artinya nilainya tidak bisa diubah setelah dibuat, struktur ini ideal untuk data yang bersifat konstan atau tetap.

Contoh Program:

```
koordinat = (10, 20)
print(koordinat[0])
# Output: 10
```

c. Set

Set merupakan kumpulan Data Unik dan Tidak Berurutan. Set menyimpan elemen unik (tidak ada duplikat) dan tidak memiliki urutan tertentu. Struktur ini efisien untuk operasi himpunan seperti union, intersection, dan difference.

Contoh Program:

```
angka = {1, 2, 3, 3, 2}
```

```
print(angka)
# Output: {1, 2, 3} set_a = {1, 2, 3}
set_b = {3, 4, 5}
print(set_a.union(set_b))
# Output: {1, 2, 3, 4, 5}
```

d. Dictionary
Dictionary merupakan struktur data yang menyimpan pasangan key–value (kunci–nilai). Setiap key harus unik dan digunakan untuk mengakses nilai dengan cepat.
Contoh Program:

```
mahasiswa = {
"nama": "Budi",
"umur": 25,
"jurusan":
"Informatika"
}
print(mahasiswa["nama"])
# Output: Budi
```

Ciri Dictionary:
a. Bersifat mutable (dapat diubah).
b. Akses data cepat berdasarkan key.
c. Ideal untuk data terstruktur seperti identitas, objek, atau konfigurasi.

9.4 Manipulasi Data pada List

Beberapa operasi umum yang dapat dilakukan pada list:

a. Menambah Elemen
```
buah.append("anggur")
buah.insert(1, "melon")
```
b. Menghapus Elemen
```
buah.remove("apel")
buah.pop(0)
```
c. Mengakses Elemen
```
print(buah[1])
print(buah[0:2])
```
d. Mengubah Elemen
```
buah[0] = "pepaya"
```

Contoh Program:

```
tinggi = [160, 170, 180]
tinggi[1] = 175
tinggi.append(190)
print(tinggi)
# [160, 175, 180, 190]
print(tinggi[1:3])  # [175, 180]
```

9.5 Perbandingan Struktur Data Python

Berikut ini perbandingan Struktur Data pada Python yang disajikan pada Tabel 9.1:

Tabel 9.1. Perbandingan Struktur Data Python

Aspek	List	Tuple	Set	Dictionary
Mutability	Mutable	Immutable	Mutable	Mutable
Urutan	Berurutan	Berurutan	Tidak Berurutan	Tidak Berurutan
Duplikat	Diizinkan	Diizinkan	Tidak Diizinkan	Tidak Diizinkan
Akses	Dengan indkes	Dengan indeks	Iterasi	Berdasarkan Key
Performa	Sedang	Cepat	Cepat	Sangat Cepat
Kegunaan	Data Dinamis	Data Konstan	Data Unik	Data Terstruktur

9.6 Pentingnya Struktur Data dalam Python

Pemilihan struktur data yang tepat memberikan dampak besar pada efisiensi program. Berikut alasan mengapa struktur data menjadi elemen krusial:

a. Pengelolaan Data Dunia Nyata, membantu mengorganisasi data dalam sistem nyata seperti e-commerce, data pelanggan, dan inventori.
b. Efisiensi & Kecepatan, meningkatkan performa akses dan manipulasi data.
c. Fondasi Algoritma, menjadi dasar dalam pembuatan algoritma sorting, searching, dan data processing.

d. Aplikasi Luas, digunakan dalam berbagai bidang seperti data science, web development, AI, dan machine learning.

Struktur data adalah jantung dari setiap program yang efisien. Menguasainya adalah langkah pertama menjadi programmer Python yang handal.

9.7 RANGKUMAN

Struktur data adalah cara Python menyimpan dan mengelola data agar mudah diproses.

a. Terdapat empat struktur data dasar yang dipelajari: List, Tuple, Set, dan Dictionary.
b. List bersifat mutable, dapat menampung berbagai tipe data, dan mendukung banyak operasi seperti menambah, menghapus, dan mengubah elemen.
c. Tuple bersifat immutable, sehingga lebih aman dan lebih ringan digunakan untuk data yang tidak boleh diubah.
d. Set menyimpan elemen unik tanpa duplikasi, tidak memiliki indeks, dan sangat cepat untuk pengecekan keanggotaan.
e. Dictionary menyimpan data dalam bentuk pasangan key–value, memudahkan pencarian dan pengelompokan informasi.
f. Setiap struktur data memiliki kelebihan dan kekurangan, sehingga pemilihannya harus disesuaikan dengan kebutuhan program.
g. Pemahaman struktur data membantu penulisan program yang lebih efisien, rapi, dan mudah dikembangkan.

BAB 10
DICTIONARY DAN SET PADA PEMROGRAMAN PYTHON

10.1 Pendahuluan

Bab ini membahas dua struktur data penting dalam Python, yaitu Dictionary dan Set. Keduanya termasuk struktur data yang menggunakan mekanisme hash table, sehingga mampu memberikan performa pencarian dan pengecekan data yang sangat cepat. Pemahaman terhadap dua struktur data ini membantu dalam pengolahan data yang kompleks serta pembuatan program yang efisien.

10.2 Pengertian Dictionary

Dictionary adalah struktur data yang menyimpan informasi dalam pasangan kunci (key) dan nilai (value). Bayangkan seperti kamus sungguhan: Anda mencari kata (key) untuk menemukan artinya (value). Keunggulan utama dictionary adalah kecepatan akses data yang sangat tinggi berkat penggunaan hash table. Contoh: {'nama': 'Andi', 'umur': 20, 'kota': 'Bandung'}

Ciri-ciri Dictionary sesuai modul:

a. Menggunakan pasangan key–value
b. Key harus unik
c. Value dapat berupa tipe data apapun
d. Ditulis menggunakan { }

10.3 Membuat Dictionary

a. Mendefinisikan dictionary secara langsung. Contoh:
```
data = {'buah': 'apel', 'jumlah': 5}
```
b. Menggunakan dict(). Contoh :
```
data = dict(nama='Budi', umur=22)
```

10.4 Mengakses dan Memodifikasi Dictionary

a. Mengakses nilai menggunakan key
```
print(data['nama'])
```
b. Menambah atau mengubah data
```
data['umur'] = 25
data['kota'] = 'Jakarta'
```
c. Menghapus data
```
del data['umur']
```

10.5 Method pada Dictionary

a. .keys(), digunakan untuk mengembalikan daftar key.
```
data.keys()
```
b. .values(), digunakan untuk mengembalikan daftar value.
```
data.values()
```
c. .items(), digunakan untuk mengembalikan pasangan key–value.
```
data.items()
```
d. .get(), digunakan untuk mengakses key secara aman.
```
data.get('nama')
```
e. .pop(), digunakan untuk menghapus key tertentu.
```
data.pop('kota')
```

10.6 Pengertian Set

Set adalah kumpulan elemen unik tanpa urutan tertentu. Mirip dengan konsep himpunan dalam matematika, set di Python tidak mengizinkan duplikasi dan tidak menjaga urutan elemen. Set sangat efisien untuk operasi himpunan seperti mencari elemen yang sama atau berbeda antar kelompok data. Ciri-ciri Set yaitu:

a. Tidak menerima nilai duplikat
b. Tidak memiliki indeks
c. Menggunakan { }
d. Sering digunakan untuk operasi himpunan

Keunggulan Set dalam pemrograman Python adalah:

a. Otomatis menghilangkan duplikasi
b. Pencarian sangat cepat (O(1))

c. Mendukung operasi matematika himpunan
Contoh:
```
angka = {1, 2, 3, 4, 5} # Duplikat otomatis
dihapus
huruf = {'a','b','a','c'} # Hasil: {'a','b','c
'}
```

10.7 Operasi Dasar pada Set

a. Menambah elemen
```
my_set = {1, 2, 3} my_set.add(6)# {1, 2, 3,
6}
```
b. Menghapus elemen
```
my_set.remove(3)# Error jika tidak ada
my_set.discard(3)# Error
```
c. Union (gabungan)
```
A = {1, 2, 3}
B = {3, 4, 5}
A | B # {1,2,3,4,5}
```
d. Intersection (irisan)
```
A & B # {3}# Elemen yang ada# di kedua set
```
e. Difference (selisih)
```
A B # {1, 2}# Elemen di A# tapi tidak di B
```

10.8 Perbedaan Utama Dictionary dan Set

a. Dictionary:
   1. Menyimpan pasangan key-value
   2. Key harus unik, value boleh duplikat
   3. Akses data menggunakan key
      Contoh: {'nama': 'Andi', 'umur': 25}
   4. Cocok untuk data terstruktur dengan label

b. Set:
   1. Hanya kumpulan elemen unik
   2. Tidak ada konsep key-value
   3. Tidak ada urutan/indeks
      Contoh: {1, 2, 3, 4, 5}
   4. Cocok untuk operasi himpunan matematika

Persamaan keduanya menggunakan hash table sehingga sangat cepat untuk operasi pencarian dan pengecekan keanggotaan – kompleksitas waktu O(1).

10.9 RANGKUMAN

a. Dictionary menyimpan data dalam pasangan key–value.
b. Set menyimpan elemen unik tanpa urutan.
c. Dictionary bersifat mutable dan key harus unik.
d. Set tidak menerima duplikasi dan cepat untuk operasi himpunan.
e. Method dictionary yang penting: keys(), values(), items(), get(), pop().
f. Operasi set yang penting: add, remove, union (|), intersection (&), difference (-).
g. Dictionary dan set sama-sama menggunakan hash sehingga aksesnya cepat.
h. Modul memberikan contoh dictionary berisi set dan penggunaan defaultdict.

BAB 11
MEMAHAMI DEFINISI, KEGUNAAN, DAN STRUKTUR FUNGSI PADA PEMROGRAMAN PYTHON

11.1 Pendahuluan

Fungsi merupakan salah satu konsep fundamental dalam pemrograman Python yang dirancang untuk menyederhanakan kode dan membuat program lebih terstruktur. Dengan fungsi, sebuah tugas tertentu dapat dipisahkan menjadi blok kode mandiri sehingga lebih mudah digunakan kembali, dipelihara, dan dikembangkan. Pemahaman fungsi menjadi langkah penting sebelum mempelajari konsep lanjutan seperti modularisasi, rekursi, dan pemrograman berorientasi objek.

11.2 Pengertian Fungsi dalam Python

Fungsi adalah blok kode yang dirancang untuk melakukan tugas tertentu, yang dapat dipanggil (call) berkali-kali dalam sebuah program. Fungsi membantu memecah program kompleks menjadi bagian-bagian kecil yang lebih mudah dipahami dan dikelola. Manfaat Penggunaan Fungsi sebagai berikut:

a. Reusability
   Kode yang sama tidak perlu ditulis berulang. Cukup definisikan fungsi, lalu panggil kapan pun dibutuhkan.
b. Struktur Jelas
   Program lebih terorganisir sehingga mudah dibaca dan dipahami oleh programmer lain.
c. Mudah Dipelihara
   Ketika terjadi kesalahan, perbaikan cukup dilakukan pada satu tempat: fungsi tersebut.

11.3 Kegunaan Fungsi

Python dan fungsi-fungsinya telah menjadi tulang punggung berbagai aplikasi modern. Mari kita jelajahi bagaimana fungsi Python digunakan dalam berbagai industri dan proyek teknologi.

a. Otomatisasi Tugas
   Fungsi Python sangat efektif untuk mengotomatisasi tugas berulang seperti pengelolaan file, pengiriman email massal, web scraping, dan backup data otomatis. Hemat waktu hingga berjam-jam setiap harinya.
b. Pengembangan Web
   Framework populer seperti Django dan Flask mengandalkan fungsi untuk membangun aplikasi web yang scalable. Dari startup hingga perusahaan besar seperti Instagram dan Spotify menggunakan Python.
c. Analisis Data dan Kecerdasan Buatan (AI)
   Pustaka seperti NumPy, Pandas, dan TensorFlow memanfaatkan fungsi untuk mengolah data besar dan membangun model machine learning. Python adalah bahasa #1 untuk data science
d. Aplikasi Desktop dan Game
   Dengan pustaka Tkinter untuk GUI dan Pygame untuk game development, fungsi Python memungkinkan pembuatan aplikasi desktop interaktif dan game 2D yang menarik.

11.4 Struktur Dasar Fungsi Python

Setiap fungsi Python memiliki komponen-komponen penting yang membentuk struktur dasarnya. Memahami struktur ini adalah kunci untuk menulis kode yang efektif.

Code Program:

```
def nama_fungsi(parameter1, parameter2):
"""Docstring: penjelasan fungsi""" # Blok kode
fungsi hasil = parameter1 + parameter2 return
hasil# Memanggil fungsioutput =
nama_fungsi(5, 3)
```

11.5 RANGKUMAN

a. Fungsi adalah blok kode yang digunakan untuk menjalankan tugas tertentu dan dapat dipanggil berkali-kali.
b. Fungsi membantu membuat program lebih terstruktur, mudah dibaca, dan lebih efisien.
c. Keuntungan utama fungsi adalah reusability, pengelolaan kode yang lebih baik, dan pemeliharaan program yang lebih mudah.
d. Fungsi sangat penting dalam berbagai bidang seperti otomatisasi, pengembangan web, analisis data, kecerdasan buatan, dan pembuatan aplikasi.
e. Struktur dasar fungsi terdiri atas: kata kunci def, nama fungsi, parameter, docstring, blok kode, dan return.
f. Pemanggilan fungsi dilakukan dengan menuliskan nama fungsi dan tanda kurung (), serta mengisi argumen jika diperlukan.
g. Fungsi mendukung pemrosesan data, pembuatan modular program, dan penerapan logika yang lebih kompleks dalam pemrograman Python.

BAB 12
STRUKTUR EKSEPSI PADA PEMROGRAMAN PYTHON

12.1 Pendahuluan

Dalam proses pemrograman, kesalahan sering terjadi baik karena penulisan kode maupun kondisi tak terduga saat program berjalan. Oleh karena itu, Python menyediakan mekanisme untuk menangani kesalahan tersebut agar program tidak berhenti tiba-tiba. Bab ini membahas konsep dasar eksepsi, perbedaan antara syntax error dan runtime error, serta cara menangani kesalahan menggunakan struktur try dan except sehingga program dapat berjalan lebih stabil dan aman.

12.2 Eksepsi dalam Python

Eksepsi adalah kesalahan yang terjadi ketika program sedang berjalan (runtime). Berbeda dengan syntax error yang muncul sebelum kode dijalankan, eksepsi terjadi meskipun kode secara penulisan sudah benar namun terdapat kondisi tak terduga saat eksekusi berlangsung. Beberapa contoh umum eksepsi dalam Python antara lain:

a. Pembagian dengan angka nol
b. Akses pada variabel yang belum didefinisikan
c. Membuka file yang tidak tersedia
d. Konversi tipe data yang tidak valid

Eksepsi tidak selalu menyebabkan program berhenti secara fatal. Dengan penanganan yang tepat, program tetap dapat berjalan dan memberikan respons yang sesuai ketika terjadi kesalahan.

12.3 Dua Jenis Kesalahan dalam Python

Dalam Python, kesalahan dibagi menjadi dua kategori utama:

a. Syntax Error
   Kesalahan yang muncul akibat kesalahan penulisan kode. Python mendeteksi kesalahan ini sebelum program dijalankan.
   Contoh:
   1. Tidak menggunakan tanda titik dua (:) pada struktur kontrol
   2. Kesalahan indentasi
   3. Tanda kurung yang tidak seimbang
   4. Penulisan keyword yang salah
b. Exception (Runtime Error)
   Kesalahan yang terjadi ketika program sedang berjalan.
   Contoh:
   1. ZeroDivisionError → pembagian dengan nol
   2. NameError → variabel belum didefinisikan
   3. TypeError → operasi dilakukan pada tipe data yang salah
   4. FileNotFoundError → file tidak ditemukan

Python menyediakan pesan traceback untuk membantu programmer mengetahui baris terjadinya kesalahan sehingga proses debugging dapat dilakukan lebih cepat dan akurat.

12.4 Cara Menangani Eksepsi: Blok Try-Except

Untuk mencegah program berhenti ketika terjadi kesalahan, Python menyediakan struktur try dan except.

a. Blok try
   Digunakan untuk menempatkan kode yang berpotensi menimbulkan error. Python terlebih dahulu mencoba menjalankan kode yang berada di dalam blok ini.
b. Blok except
   Jika terjadi eksepsi pada blok try, eksekusi akan langsung berpindah ke blok except. Blok ini digunakan untuk menangani kesalahan dan memberikan respon yang sesuai.

Dengan penerapan struktur ini, program dapat tetap melanjutkan proses meskipun terjadi error.

12.5 Contoh Implementasi Penanganan Eksepsi

Berikut adalah contoh penanganan eksepsi menggunakan try dan except:

Code Program:

```
try:
hasil = 10 / 0
except ZeroDivisionError:
print("Error: Tidak bisa membagi dengan nol!")
hasil = None
print("Program tetap berjalan...")
```

Dengan struktur tersebut, ketika terjadi kesalahan pembagian dengan nol, program tidak akan crash dan tetap melanjutkan eksekusi setelah menampilkan pesan error.

12.6 RANGKUMAN

a. Eksepsi adalah kesalahan yang terjadi saat program berjalan (runtime).
b. Berbeda dengan syntax error yang muncul sebelum program dijalankan.
c. Contoh eksepsi: pembagian dengan nol, variabel tidak ada, file tidak ditemukan, konversi tipe data tidak valid.
d. Python memiliki dua jenis kesalahan: syntax error dan exception.
e. Penanganan eksepsi menggunakan blok try untuk kode yang berpotensi error.
f. Blok except digunakan untuk menangkap dan menangani error agar program tidak berhenti.
g. Dengan penanganan eksepsi, program menjadi lebih aman, stabil, dan tidak mudah crash.

DAFTAR PUSTAKA

Downey, A. (2015). *Think Python: How to Think Like a COmputer Scientist*. Green Tea Press.

Rahman, S., Sembiring, A., Siregar, D., Khair, H., Prahmana, I. G., Puspadini, R., & Zen, M. (2023). *PYTHON : DASAR DAN PEMROGRAMAN*. CV Tahta Media Group.

Romzi, M., & Kurniawan, B. (2020). Implementasi Pemrograman Python Menggunakan Visual Studio Code. *JIK*, *11*(2), 1–9.

Santoso, J. Te. (2022). *Proyek Coding dengan Python*. Yayasan Prima Agus Teknik.

Septian, R. F. (2013). *Belajar Pemrograman Python Dasar*. POSS.

Suharto, A. (2023). *Fundamental Bahasa Pemrograman Python*. EUREKA MEDIA AKSARA.

Surbakti, N. M., Angelyca, A., Talia, A., Perangin-Angin, C. B., Nainggolan, D. O., Friskauly, N. D., & Tumorang, S. R. B. (2024). Penggunaan Bahasa Pemrograman Python dalam Pembelajaran Kalkulus Fungsi Dua Variabel. *Algoritma : Jurnal Matematika, Ilmu Pengetahuan Alam, Kebumian Dan Angkasa*, *2*(3), 98–107. https://doi.org/10.62383/algoritma.v2i3.67

BIODATA PENULIS

Riki Winanjaya, M.Kom
Dosen Program Studi Manajemen Informatika
STIKOM TUNAS BANGSA

Riki Winanjaya, S.Kom, M.Kom dilahirkan di Pematangsiantar, Sumatera Utara pada tanggal 04 September 1985. Penulis menempuh pendidikan D3 dan S1 di STIKOM Tunas Bangsa bidang Ilmu Komputer dan S2 di Universitas Putra Indonesia Yptk Padang Sumatera Barat tahun 2019.

Menetap di kota Pematangsiantar dan menjadi dosen tetap di STIKOM Tunas Bangsa Program Studi Manajemen Informatika. Aktif mengajar di STIKOM Tunas Bangsa sejak tahun 2019, Aktif menulis naskah kedalam bentuk jurnal penelitian dan jurnal pengabdian yang terakreditasi oleh Sinta. Saat ini Penulis menjabat sebagai Wakil Ketua II STIKOM Tunas Bangsa Pematangsiantar

BIODATA PENULIS

Bunga Sabila
STIKOM TUNAS BANGSA

Penulis merupakan anak keempat dari empat bersaudara. Menempuh pendidikan menengah kejuruan di SMK Satrya Budi Karang Rejo dengan jurusan Rekayasa Perangkat Lunak, dan pendidikan tinggi Program Studi Sistem Informasi di STIKOM Tunas Bangsa Pematangsiantar. Penulis memiliki minat pada bidang Sistem Informasi, pengelolaan basis data, serta pengembangan aplikasi berbasis web. Penulis juga menyukai kegiatan menulis artikel maupun buku ajar serta aktif mengikuti berbagai kegiatan seminar dan workshop sebagai upaya pengembangan wawasan dan kompetensi akademik.

BIODATA PENULIS

P.P.P.A.N.W.Fikrul Ilmi R.H.Zer, M.Kom
Dosen Program Studi Manajemen Informatika
STIKOM TUNAS BANGSA

P.P.P.A.N.W.Fikrul Ilmi R.H.Zer, S.Kom, M.Kom dilahirkan di Pematangsiantar, Sumatera Utara pada tanggal 09 Juli 1997. Penulis menempuh pendidikan S1 di STIKOM Tunas Bangsa bidang Ilmu Komputer tahun 2015 dan S2 di Universitas Potensi Utama di Medan tahun 2021.

Menetap di kota Pematangsiantar dan menjadi dosen tetap di STIKOM Tunas Bangsa homebase Program Studi Manajemen Informatika. Aktif mengajar di STIKOM Tunas Bangsa sejak tahun 2023. Bidang keilmuan penulis yaitu Deep Learning dan Programming. Aktif menulis naskah kedalam bentuk jurnal penelitian dan jurnal pengabdian yang terakreditasi oleh Sinta. Saat ini Penulis menjabat sebagai Sekretaris Program Studi Teknik Informatika di STIKOM Tunas Bangsa.

BIODATA PENULIS

Widodo Saputra, M.Kom
Dosen Program Studi Manajemen Informatika
STIKOM TUNAS BANGSA

Widodo Saputra, S. Kom., M. Kom lahir di Kota Pematangsiantar, Sumatera Utara pada tanggal 01 April 1989. Penulis menempuh pendidikan Diploma 3 (D3) di AMIK Tunas Bangsa pada tahun 2007 Bidang Ilmu Komputer. Pada tahun 2012 penulis melanjutkan tingkat Sarjana (S1) di STTP Medan dan S2 di Universitas Sumatera Utara (USU) Medan pada tahun 2015.

Tinggal di Kota Pematangsiantar dan menjadi dosen tetap di STIKOM Tunas Bangsa Pematangsiantar Homebase Manajemen Informatika sejak tahun 2017. Saat ini memiliki kepangkatan Lektor dengan Golongan III/C. Bidang Ilmu penulis yaitu Pemrograman dan Artificial Inteligent. Penulis aktif menulis naskah dan publish dalam bentuk jurnal penelitian dan jurnal pengabdian terakreditasi. Saat ini penulis menjabat sebagai Sekretaris Program Studi Magister Informatika di STIKOM Tunas Bangsa

BIODATA PENULIS

Masri Wahyni, M. Kom
Dosen Program Studi Komputerisasi Akuntansi
Akademi Manajemen Informatika dan Komputer Polibisnis

Masri Wahyuni, S,Kom, M.Kom dilahirkan di Sidomulyo, Sumatera Utara pada tanggal 06 September 1983, Penulis menempuh pendidikan S1 di AKAKOM Yogyakarta bidang ilmu Sistem Informasi tahun 2001 dan S2 di Universitas Potensi Utama di Medan tahun 2021.

Menetap di kota Kisaran dan menjadi dosen tetap di AMIK Polibisnis Perdagangan homebase Program Studi Komputerisasi Akuntansi. Aktif mengajar di AMIK Polibisnis sejak tahun 2023. Bidang keilmuan penulis yaitu Ilmu Komputer. Aktif menulis naskah kedalam bentuk jurnal penelitian dan jurnal pengabdian yang terakreditasi oleh Sinta. Saat ini Penulis menjabat sebagai Wakil Direktur II AMIK Polibisnis Perdagangan.