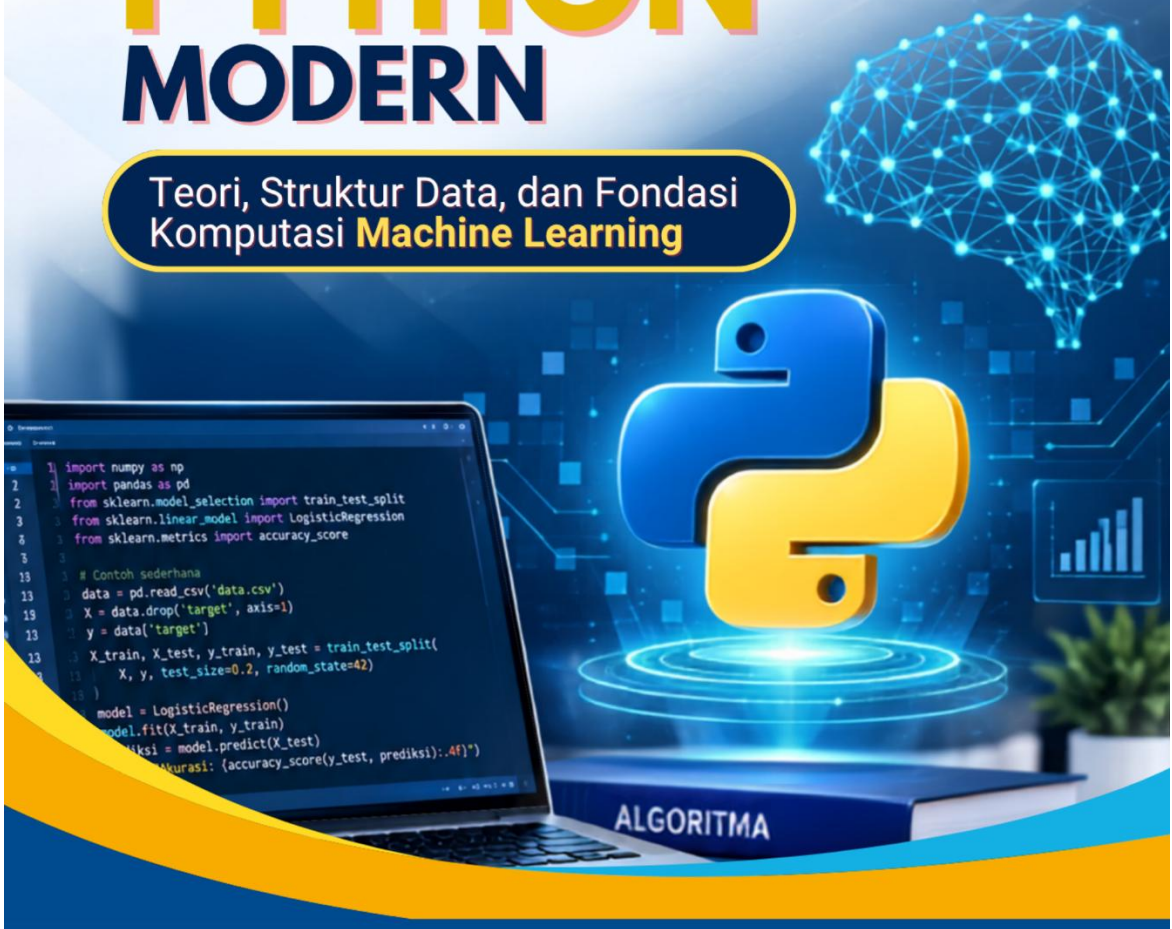
sada
kurnia pustaka

ALGORITMA DAN PEMROGRAMAN PYTHON MODERN

Teori, Struktur Data, dan Fondasi Komputasi Machine Learning



Tim Penulis:
Nina Rahayu | Lindung Siswanto | Deden Rustiana | Diki Arisandi Fahmi | Tugiman | Ahmad Budi Trisnawan | Eza Budi Perkasa Saryani | Indo Intan | Muhamad Yusup |Dewi Immaniar Desrianti Riri Fajriah | Oleh Soleh | Falahah

Editor: Ajay Supriadi

ALGORITMA DAN PEMROGRAMAN PYTHON MODERN

Teori, Struktur Data, dan Fondasi Komputasi Machine Learning

Nina Rahayu
Lindung Siswanto
Deden Rustiana
Diki Arisandi
Fahmi
Tugiman
Ahmad Budi Trisnawan
Eza Budi Perkasa
Saryani
Indo Intan
Muhamad Yusup
Dewi Immaniar Desrianti
Riri Fajriah
Oleh Soleh
Falahah

Editor: **Ajay Supriadi, M.Kom.**

sada
kurnia pustaka

ALGORITMA DAN PEMROGRAMAN PYTHON MODERN

Teori, Struktur Data, dan Fondasi Komputasi Machine Learning

Tim Penulis:
Nina Rahayu
Lindung Siswanto
Deden Rustiana
Diki Arisandi
Fahmi
Tugiman
Ahmad Budi Trisnawan
Eza Budi Perkasa
Saryani
Indo Intan
Muhamad Yusup
Dewi Immaniar Desrianti
Riri Fajriah
Oleh Soleh
Falahah

Editor : Ajay Supriadi, M.Kom.
Tata Letak : Lilis Khalisatul Karimah, S.H.
Desain Cover : Asep Nugraha, S.Hum.
Ukuran : UNESCO 15,5 x 23 cm
Halaman : ix, 295
ISBN : 978-634-7949-01-1
Terbit Pada : September 2026
Anggota IKAPI : No. 073/BANTEN/2023

PENERBIT PT SADA KURNIA PUSTAKA
Jl. Kramat, Panenjoan Kec. Carenang, Kab. Serang – Banten, 42195
Email : sadapenerbit@gmail.com
Website : sadapenerbit.com & repository.sadapenerbit.com
Telpon/WA : +62 838 1281 8431

KATA PENGANTAR

Puji syukur kita panjatkan ke hadirat Tuhan Yang Maha Esa, karena atas rahmat dan karunia-Nya, buku berjudul **"Algoritma dan Pemrograman Python Modern: Teori, Struktur Data, dan Fondasi Komputasi *Machine Learning*"** ini dapat diselesaikan dan hadir di hadapan para pembaca.

Perkembangan teknologi komputasi dalam satu dekade terakhir telah mengubah lanskap ilmu pengetahuan secara fundamental. Python, sebagai salah satu bahasa pemrograman yang paling pesat berkembang, telah menjadi *lingua franca* bagi para ilmuwan data, pengembang perangkat lunak, dan peneliti di seluruh dunia. Kehadiran Python tidak hanya mempermudah proses pembuatan program, tetapi juga menjadi jembatan antara konsep algoritma klasik dengan kebutuhan komputasi modern, terutama di bidang *machine learning* dan kecerdasan artifisial.

Buku ini disusun dengan kesadaran bahwa pembelajaran algoritma dan pemrograman tidak cukup hanya berhenti pada sintaksis bahasa. Lebih dari itu, pembaca perlu memahami bagaimana sebuah algoritma dirancang, bagaimana data disimpan dan dimanipulasi melalui struktur data yang tepat, serta bagaimana fondasi komputasi tersebut menjadi pijakan dalam membangun model *machine learning* yang efisien dan akurat.

Oleh karena itu, buku ini dibagi ke dalam tiga pilar utama. Pertama, pembahasan teori algoritma dan pemrograman Python modern yang mencakup dasar-dasar logika, kontrol alur, fungsi, hingga paradigma pemrograman berorientasi objek dan fungsional. Kedua, eksplorasi struktur data (mulai dari senarai, tumpukan, antrean, pohon, graf, hingga struktur data lanjutan) yang disajikan dengan implementasi langsung dalam Python. Ketiga, fondasi komputasi untuk *machine learning*, yang membahas dasar aljabar linear, statistika komputasi, vektorisasi, serta pengenalan *library* kunci seperti NumPy, pandas, dan scikit-learn.

Akhir kata, kami berharap buku ini dapat menjadi teman belajar yang bermanfaat bagi mahasiswa, dosen, praktisi, maupun siapa pun yang ingin menyelami dunia algoritma, Python, dan *machine learning* secara lebih mendalam.

Selamat membaca dan selamat berkarya.

Penulis

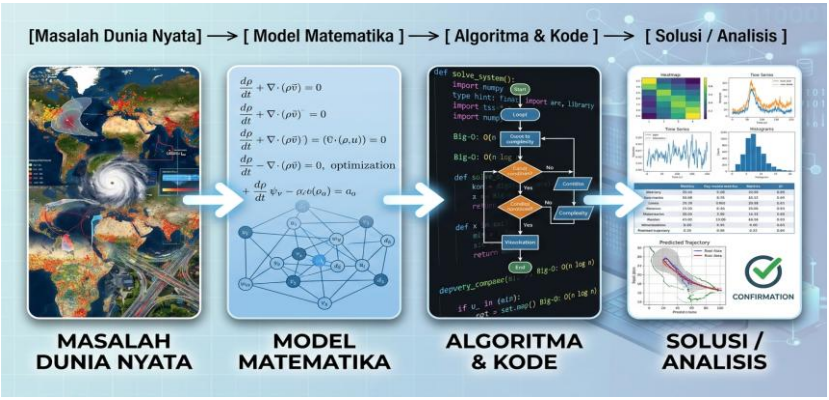
DAFTAR ISI

BAB 1
PENGANTAR ILMU KOMPUTASI DAN PYTHON

Nina Rahayu, S.Kom., M.M., M.T.I.
Universitas Raharja

Paradigma Komputasi Sains dan Pemikiran Algoritmik

Dalam lanskap ilmu pengetahuan modern, ilmu komputasi (*computational science*) telah diakui secara luas sebagai pilar ketiga dalam metodologi eksplorasi ilmiah, melengkapi dan memperkuat dua pilar konvensional: pendekatan teori (*theoretical*) dan eksperimen laboratorium (*experimental*) (Lu et al., 2021). Komputasi sains bukan sekadar penggunaan perangkat komputer untuk memproses angka, melainkan integrasi multidisiplin antara matematika terapan, ilmu komputer, dan domain spesifik (seperti fisika, biologi, ekonomi, hingga kecerdasan buatan) untuk mensimulasikan serta memprediksi fenomena kompleks yang sulit diuji secara langsung di alam nyata.



Gambar 1.1: Tahapan Pemecahan Masalah dan Pemodelan dalam Ilmu Komputasi

Sumber: Diadaptasi dari Shiflet & Shiflet, 2014; Wing, 2006)

Inti dari ilmu komputasi terletak pada kemampuan menerapkan pemikiran algoritmik atau *computational thinking* (Wing, 2006). Pemikiran algoritmik adalah proses mental dalam memformulasi suatu masalah sedemikian rupa sehingga solusinya dapat direpresentasikan dalam bentuk langkah-langkah komputasi yang dapat dieksekusi oleh mesin. Terdapat empat elemen kunci dalam pemikiran algoritmik:

1. Dekomposisi (*Decomposition*): Memecah masalah kompleks yang berskala besar menjadi bagian-bagian terkecil yang lebih mudah dikelola dan dianalisis.

2. Pengenalan Pola (*Pattern Recognition*): Mengidentifikasi persamaan, tren, atau karakteristik berulang dari data atau masalah untuk menentukan pendekatan komputasi yang tepat.
3. Abstraksi (*Abstraction*): Mengeliminasi rincian yang tidak relevan dan berfokus hanya pada informasi atau konsep inti yang esensial untuk pemodelan.
4. Rancangan Algoritma (*Algorithm Design*): Menyusun instruksi matematis dan logis yang terstruktur secara berurutan (*step-by-step*) untuk menyelesaikan masalah.

Dalam konteks *machine learning* (ML) dan sains data, pemikiran algoritmik bertindak sebagai jembatan antara persoalan matematis (seperti minimasi fungsi kerugian (*loss function*)) dan eksekusi sistem komputer secara efisien. Tanpa fondasi algoritmik dan pemilihan struktur data yang optimal, metode komputasi yang diterapkan akan menghadapi kendala kompleksitas waktu dan eksploitasi memori (*computational complexity*).

Peran dan Evolusi Python dalam Lanskap Komputasi Modern

Diciptakan oleh Guido van Rossum pada tahun 1991, Python awalnya dikembangkan sebagai bahasa pemrograman tingkat tinggi yang berfokus pada keterbacaan kode dan kemudahan penggunaan. Namun, dalam kurun waktu dua dekade terakhir, Python telah mengalami evolusi mendasar, transformasi dari bahasa skrip sederhana menjadi standar industri mutlak dalam pemrosesan data, komputasi ilmiah, dan kecerdasan buatan (Blayo et al., 2023).

Keunggulan dominan Python dalam komputasi modern didasari oleh filosofi desainnya yang terangkum dalam *The Zen of Python* (PEP 20), yang menekankan kesederhanaan, transparansi, dan efisiensi penulisan kode. Berdasarkan analisis komprehensif mengenai bahasa komputasi berkinerja tinggi, dominasi Python didorong oleh faktor-faktor teknis utama berikut:

1. Karakteristik Teknis: Deskripsi dan Dampaknya pada Komputasi Ilmiah.
2. Sintaksis Bersih dan Keterbacaan Tinggi: Meminimalkan *cognitive load* peneliti dan praktisi, memungkinkan fokus penuh dialihkan pada logika algoritma ketimbang kerumitan sintaksis.

Daftar Pustaka

Blayo, J., et al. (2023). Landscape of High-Performance Python to Develop Data Science and *Machine learning* Applications. *ACM Computing Surveys*, 56(3), 1–30. https://doi.org/10.1145/3617588

Harris, C. R., Millman, K. J., van der Walt, S. J., et al. (2020). Array programming with NumPy. *Nature*, 585(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Lu, L., Meng, X., Mao, Z., & Karniadakis, G. E. (2021). DeepXDE: A deep learning library for solving differential equations. *Nature Computational Science*, 63(1), 69–78. https://doi.org/10.1137/19M1274067

McKinney, W. (2010). Data structures for statistical *computing* in Python. *Proceedings of the 9th Python in Science Conference (SciPy 2010)*, 51–56.

Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). Scikit-learn: *Machine learning* in Python. *Journal of Machine learning Research*, 12, 2825–2830.

Virtanen, P., Gommers, R., Oliphant, T. E., et al. (2020). SciPy 1.0: fundamental algorithms for scientific *computing* in Python. *Nature Methods*, 17(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33–35. https://doi.org/10.1145/1118178.1118215

Perkel, J. M. (2018). Why Jupyter is data scientists' computational notebook of choice. *Nature*, 563, 145–146. *https://doi.org/10.1038/d41586-018-07196-1*

Rule, A., Tabard, A., & Hollan, J. D. (2018). Exploration and explanation in computational notebooks. In Proceedings of the 2018 CHI Conference on Human Factors in *Computing* Systems (1–12). Association for *Computing* Machinery. https://doi.org/10.1145/3173574.3173606

PROFIL PENULIS

Nina Rahayu, S. Kom., M.M., M.T.I.

Penulis adalah seorang akademisi dan peneliti di bidang teknologi informasi dengan pengalaman lebih dari satu dekade. Saat ini, penulis berperan sebagai dosen di Universitas Raharja sejak tahun 2016. Dengan latar belakang pendidikan di bidang Manajemen dan Teknologi Informasi, penulis juga aktif dalam pengajaran, penelitian, dan pengembangan sistem berbasis teknologi.

Selain mengajar dan meneliti, penulis juga memiliki pengalaman dalam implementasi sistem teknologi di industri, seperti keterlibatannya dalam proyek pada bagian ITM di PT. Telkomsigma. Komitmennya terhadap dunia akademik dibuktikan dengan kelanjutan studinya pada program doktoral di Universitas Kristen Satya Wacana, di mana ia terus mengembangkan penelitian di bidang ilmu komputer. Sebagai seorang peneliti, penulis telah menulis berbagai artikel ilmiah dan berpartisipasi dalam hibah penelitian, termasuk proyek-proyek seperti prediksi biaya kuliah menggunakan kecerdasan buatan, sistem peringatan dini kebakaran berbasis sensor fusion, dan pengembangan sistem keamanan data berbasis *blockchain* untuk *e-voting*.

Dengan dedikasi yang tinggi dalam dunia pendidikan dan teknologi, penulis terus berkontribusi dalam inovasi sistem informasi dan kecerdasan buatan, serta berperan dalam mencetak generasi muda yang siap menghadapi tantangan digital.

Email Penulis: ninarahayu.niez@gmail.com

BAB 2
SINTAKS DASAR DAN TIPE DATA

Lindung Siswanto, S.Kom., M.Eng.
Politeknik Negeri Pontianak

Sintaks Python

Sintaks merupakan aturan penulisan yang harus dipatuhi oleh penulis program. Setiap bahasa pemrograman memiliki aturan penulisan masing-masing. Jika aturan penulisan tersebut dilanggar, akan ditampilkan pesan *syntax error*. Python memiliki sintaks yang relatif sederhana dan menitikberatkan pada keterbacaan (*readability*).
Contoh:

```
1   print("Hello, World!")
```

Kode di atas sudah mengikuti sintaks yang benar. Jika kode tersebut dijalankan, akan menampilkan tampilan seperti berikut:

```
$ python hello.py
Hello, World!
```

Untuk menjalankan kode Python, dilakukan dengan mengetikkan:

$ python nama_file.py atau $ py nama_file.py

Python akan menampilkan pesan kesalahan jika kode tidak sesuai dengan aturan. Berikut contohnya:

```
1   print("Hello, World!"
```

Jika dijalankan, maka akan muncul pesan seperti berikut:

```
    print("Hello, World!"
          ^
SyntaxError: '(' was never closed
```

Pesan error tersebut muncul karena tanda kurung buka "(" pada fungsi print belum ditutup dengan tanda kurung tutup ")".
Aturan dasar penulisan kode Python:

1. **Indentasi**

   Python tidak menggunakan kurawal "{}" atau "begin/end" untuk menandai blok kode, baik itu fungsi, percabangan, atau perulangan. Python menggunakan indentasi sebagai penanda blok kode. Aturan umum penulisan indentasi:

   a. Menggunakan 4 spasi per tingkat indentasi
   b. Hindari menggabungkan spasi dan tab dalam satu file.

   Berikut contoh penulisan kode yang benar dan kode yang salah beserta *output*nya:

Kode	*Output*

`# Benar`
`total=15`
`if total > 10:`
`  print("Besar")`
`  print("Selesai")`
	`$ py indentasi.py`
	`Besar`
	`Selesai`

`# Salah`
`total=15`
`if total > 10:`
`print("Besar")`
`print("Selesai")`
	`print("Besar")`
	`^^^^^`
	`IndentationError: expected an indented block after 'if' statement on line 3`

2. Pergantian baris sebagai akhir pernyataan

Python tidak menggunakan semicolon ";" di akhir baris sebagai tanda akhir suatu pernyataan seperti dalam bahasa C, Java atau PHP. Setiap baris dianggap sebagai akhir pernyataan perintah. Berikut contoh pergantian baris pada Python:

Kode	*Output*

`nama = "Budi"`
`usia = 25`
`print("Nama", nama, "usia", usia, "tahun.")`
	`$ py akhir_baris.py`
	`Nama Budi usia 25 tahun.`

`nama = "Budi" usia = 25`
`print("Nama", nama, "usia", usia, "tahun.")`
	`nama = "Budi" usia = 25`
	`^^^^`
	`SyntaxError: invalid syntax`

`nama = "Budi"; usia = 25` `print("Nama", nama, "usia", usia, "tahun.")`	`$ py akhir_baris.py` `Nama Budi usia 25 tahun.`

Contoh pada kode ketiga bisa berjalan seperti kode pertama, tetapi tidak disarankan karena tidak "Pythonic". Pythonic adalah gaya penulisan kode yang mengikuti prinsip, filosofi dan idiom bahasa pemrograman Python yang dinilai bersih, mudah dibaca, ringkas dan memanfaatkan fitur bawaan Python secara maksimal.

3. *Case Sensitivity*

Python membedakan penulisan huruf besar dan huruf kecil (*case sensitivity*). Contoh pada penulisan variabel, penulisan nama, nama atau NAMA dianggap entitas yang berbeda sepenuhnya.

Daftar Pustaka

Python Software Foundation. (n.d.). *The Python language reference*. Retrieved September 6, 2026, from https://docs.python.org/3/reference/

Van Rossum, G., Warsaw, B., & Coghlan, A. (2001). *PEP 8 – Style guide for Python code*. Python Software Foundation. https://peps.python.org/pep-0008/

PROFIL PENULIS

Lindung Siswanto, S.Kom., M.Eng.

Merupakan salah satu dosen tetap di Politeknik Negeri Pontianak dengan keahlian di bidang pemrograman web, *cloud computing*, dan sistem keamanan informasi. Menyelesaikan pendidikan Sarjana Komputer di Sekolah Tinggi Teknologi Informasi Respati Yogyakarta (kini Universitas Respati Yogyakarta), kemudian melanjutkan studi Magister Teknologi Informasi di Universitas Gadjah Mada, Yogyakarta, dengan fokus pada pengembangan sistem informasi dan teknologi keamanan.

Saat ini aktif mengembangkan metode pembelajaran berbasis praktik, menyusun modul dan bahan ajar, serta membimbing mahasiswa dalam berbagai proyek teknologi informasi, khususnya di bidang pengembangan aplikasi web dan sistem berbasis *cloud*. Secara rutin terlibat sebagai pembicara seminar dan pelatihan yang membahas topik web modern, keamanan siber, serta integrasi teknologi *cloud* dalam sistem informasi.

Selain kegiatan pengajaran, ia juga aktif dalam penelitian dan pengabdian kepada masyarakat di bidang teknologi informasi, dengan fokus pada peningkatan literasi digital, keamanan data, dan penerapan teknologi informasi di lingkungan pendidikan serta industri lokal.

Untuk mendukung kompetensi profesional, telah memiliki sejumlah sertifikasi internasional, antara lain Certified Web Developer (CWDEV), Certified Ethical Hacker (CEH), dan Certified Network Defender (CND), yang memperkuat keahlian di bidang pengembangan aplikasi web yang aman, pengujian keamanan, serta pertahanan jaringan komputer. Email Penulis: lindung_siswanto@polnep.ac.id

BAB 3
STRUKTUR KONTROL LANJUTAN

Deden Rustiana, M. Kom.
Universitas Raharja

Pendahuluan: Dari Kontrol Dasar ke Kontrol Lanjutan

Pada bab sebelumnya, tiga struktur kontrol fundamental telah dibahas: sekuens, seleksi, dan iterasi. Berdasarkan Teorema Böhm–Jacopini (1966), ketiga konstruksi ini terbukti secara teoritis sudah cukup untuk menyatakan seluruh algoritma yang dapat dikomputasi oleh mesin Turing. Secara formal, seorang pemrogram tidak memerlukan konstruksi lain di luar `if` dan `while` untuk menyelesaikan permasalahan komputasi apa pun.

Namun, kecukupan teoritis tidak selalu berbanding lurus dengan kecukupan praktis dalam rekayasa perangkat lunak dan komputasi ilmiah modern. Kode yang ditulis hanya menggunakan konstruksi minimal cenderung menjadi panjang, sulit dibaca (*unreadable*), rawan kesalahan (*error-prone*), serta kompleks untuk dipelihara (Oliveira et al., 2020). Oleh karena itu, bahasa pemrograman modern seperti Python menyediakan struktur kontrol lanjutan. Konstruksi ini tidak menambah daya komputasi (*computational power*), melainkan meningkatkan daya ekspresi (*expressive power*), keandalan eksekusi, dan keterbacaan kode (*code readability*) (Prechelt, 2000; Blayo et al., 2023).

Penggunaan fitur kontrol lanjutan seperti *comprehensions*, *generator pipelines*, dan *context managers* secara signifikan mengurangi beban kognitif pemrograman serta meminimalkan potensi kesalahan alokasi memori saat menangani dataset berskala besar (Rule et al., 2018; Blayo et al., 2023). Peta perbandingan antara paradigma kontrol dasar dan lanjutan dirangkum dalam Tabel 3.1 berikut:

Tabel 3.1: Perbandingan Karakteristik Struktur Kontrol Dasar dan Lanjutan dalam Python.

Aspek	Struktur Kontrol Dasar	Struktur Kontrol Lanjutan
Fokus	*Bagaimana* komputasi dilakukan secara eksplisit	*Apa* yang ingin dihasilkan (deklaratif)
Paradigma	Imperatif eksplisit	Deklaratif / Fungsional idiomatis

Pengelolaan State	Manual (variabel akumulator, indeks manual)	Implisit (protokol iterasi dan generator)
Konsumsi Memori	Sering kali *eager* (seluruh data dimuat ke RAM)	Dapat *lazy* (dihitung secara *on-demand*)
Risiko Galat	Tinggi (*off-by-one error*, *infinite loop*)	Lebih rendah (struktur menjamin invarian)
Contoh Konstruksi	while, if-elif-else	comprehension, generator, match, with

Sumber: Diadaptasi dari Böhm & Jacopini, 1966; Prechelt, 2000; Oliveira et al., 2020; Blayo et al., 2023).

Percabangan Lanjutan & Evaluasi Pintas (*Short-Circuit Evaluation*)

Meskipun struktur percabangan dasar `if-elif-else` mampu menangani logika keputusan sederhana, penulisan percabangan bersarang (*nested conditionals*) yang terlalu dalam kerap memicu kompleksitas siklomatis (*cyclomatic complexity*) yang tinggi. Hal ini berbanding lurus dengan peningkatan beban kognitif pengembang dan munculnya bug saat pemeliharaan perangkat lunak (Oliveira et al., 2020). Untuk mengatasi permasalahan tersebut, Python menyediakan mekanisme evaluasi ekspresi logika lanjutan serta konstruksi sintaksis yang lebih ringkas dan idiomatis.

1. **Mekanisme *Short-Circuit Evaluation* dan Nilai *Truthy/Falsy***

   Di dalam Python, operator logika `and` dan `or` tidak sekadar mengembalikan nilai kebenaran Boolean murni (`True` atau `False`), melainkan mengembalikan salah satu nilai operan yang dievaluasi. Proses evaluasi ini dihentikan segera setelah hasil akhir logika dapat dipastikan tanpa perlu mengevaluasi operan berikutnya, suatu mekanisme yang dikenal sebagai Evaluasi Pintas (*Short-Circuit Evaluation*).

   Aturan kerja evaluasi pintas ditentukan berdasarkan bobot kebenaran (*truthiness*) dari setiap objek di Python:

Daftar Pustaka

Blayo, J., et al. (2023). Landscape of High-Performance Python to Develop Data Science and *Machine learning* Applications. *ACM Computing Surveys*, 56(3), 1–30. https://doi.org/10.1145/3617588 [Google Scholar]

Böhm, C., & Jacopini, G. (1966). Flow diagrams, turing machines and languages with only two formation rules. *Communications of the ACM*, 9(5), 366–371. https://doi.org/10.1145/355592.365646 [Google Scholar]

Erich, G., Richard, H., Ralph, J., John, V. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley. [Google Scholar]

Oliveira, Delamaro, M. E., & Vincenzi, A. M. R. (2020). Code readability and maintainability metrics: A systematic mapping study. *Science of Computer Programming*, 198, 102511. https://doi.org/10.1016/j.scico.2020.102511 [Google Scholar]

Prechelt, L. (2000). An empirical comparison of seven programming languages. *IEEE Computer*, 33(10), 23–29. https://doi.org/10.1109/2.876288 [Google Scholar]

Python Software Foundation. (2024). *Python Language Reference, version 3.12 (Boolean operations)*. https://docs.python.org/3/reference/expressions.html#boolean-operations

Ramalho, L. (2022). *Fluent Python: Clear, Concise, and Effective Programming* (2nd ed.). O'Reilly Media. [Google Scholar]

Rule, A., Tabard, A., & Hollan, J. D. (2018). Exploration and explanation in computational notebooks. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (pp. 1–12). ACM. https://doi.org/10.1145/3173574.3173606 [Google Scholar]

PROFIL PENULIS

Deden Rustiana, M. Kom.

Penulis merupakan dosen tetap di Universitas Raharja sejak tahun 2016, dengan fokus pada pengajaran dan pengembangan keilmuan di bidang Teknologi Informasi. Beliau mengampu berbagai mata kuliah seperti Sistem Informasi, Pemrograman, Database, serta Manajemen Proyek TI. Selain aktif dalam dunia akademik, Deden juga berkiprah sebagai konsultan IT untuk berbagai perusahaan swasta dan instansi pemerintahan. Pengalaman konsultasinya mencakup pengembangan sistem informasi, audit TI, transformasi digital, serta integrasi teknologi berbasis web dan mobile. Dengan latar belakang pendidikan Magister Ilmu Komputer dari Universitas Budi Luhur, Deden Rustiana terus berkontribusi dalam pengembangan keilmuan TI, baik melalui pengajaran, penelitian, maupun implementasi langsung di dunia industri dan pemerintahan. Email Penulis: deden.rustiana@raharja.info

BAB 4
STRUKTUR DATA FUNDAMENTAL PYTHON

Diki Arisandi, S.Kom., M.Kom., Ph.D.
Universitas Muhammadiyah Riau

Struktur data merupakan komponen fundamental dalam pemrograman yang berperan dalam mengorganisasikan dan mengelola data agar dapat disimpan, diakses, serta dimanipulasi secara efektif. Pemilihan struktur data berkaitan erat dengan perancangan algoritma karena menentukan cara data direpresentasikan dan dioperasikan. Python menyediakan berbagai struktur data bawaan, di antaranya *list, tuple, set,* dan *dictionary*, yang memiliki karakteristik, aturan penggunaan, serta operasi yang berbeda. Pemahaman terhadap struktur tersebut penting dalam pemrograman sekaligus menjadi dasar bagi pengolahan dan transformasi data dalam berbagai bidang komputasi, termasuk analisis data dan *machine learning*. Oleh karena itu, bab ini membahas struktur data fundamental Python secara sistematis, meliputi konsep dasar, karakteristik, operasi, dan penerapannya dalam penyelesaian permasalahan pemrograman.

Pendahuluan Struktur Data Python

Python menyediakan mekanisme yang fleksibel untuk merepresentasikan berbagai bentuk data dalam suatu program. Data dapat direpresentasikan sebagai nilai tunggal maupun sebagai kumpulan elemen yang memiliki hubungan tertentu. Dalam penggunaannya, kumpulan data tersebut dapat diperlakukan sebagai satu kesatuan sehingga proses pemrograman, seperti pengulangan, pencarian, pengurutan, dan transformasi data, dapat dilakukan dengan lebih terstruktur (Shi, 2025). Kemampuan Python dalam menyediakan berbagai bentuk representasi data tersebut menjadi salah satu karakteristik yang mendukung penggunaannya dalam beragam bidang komputasi (Castro et al., 2024).

Struktur data bawaan Python memiliki cara kerja dan karakteristik yang berbeda. Perbedaan tersebut dapat dilihat dari bagaimana elemen disusun, bagaimana data diakses, apakah isi struktur dapat diubah setelah dibuat, serta apakah suatu struktur mengizinkan adanya elemen yang sama. Sebagai contoh, suatu data dapat membutuhkan akses berdasarkan posisi elemen, sementara data lainnya lebih tepat diakses berdasarkan suatu kunci tertentu.

Oleh karena itu, pemahaman terhadap karakteristik setiap struktur data diperlukan agar penggunaannya dapat disesuaikan dengan bentuk data dan operasi yang dibutuhkan dalam suatu program (Unpingco, 2021).

Dalam pembahasan struktur data fundamental Python, perhatian utama diarahkan pada empat struktur data bawaan yang umum digunakan, yaitu *list, tuple, set*, dan *dictionary*. Keempatnya memiliki bentuk representasi dan mekanisme pengelolaan data yang berbeda. Pemahaman terhadap perbedaan tersebut menjadi dasar untuk menentukan cara yang tepat dalam menyimpan dan memproses data pada program Python (Muddana & Vinayakam, 2024b). Pembahasan berikutnya akan menguraikan karakteristik dan penggunaan masing-masing struktur data tersebut secara lebih rinci.

Konsep Dasar Struktur Data

Dalam pemrograman, struktur data digunakan untuk merepresentasikan kumpulan data beserta hubungan antarelemen di dalamnya (Khemani et al., 2024). Cara suatu data direpresentasikan akan memengaruhi bagaimana data tersebut dapat diakses, ditambahkan, diubah, maupun dihapus. Oleh karena itu, struktur data tidak hanya dipahami sebagai tempat untuk menyimpan data, tetapi juga sebagai mekanisme yang menentukan bagaimana data dapat digunakan dalam suatu algoritma. Python menyediakan struktur data bawaan dengan karakteristik yang berbeda sehingga programmer dapat memilih representasi yang sesuai dengan kebutuhan pengolahan data (Abdul Kadhar & Anand, 2021).

Salah satu aspek penting dalam memahami struktur data Python adalah urutan elemen. Beberapa struktur data mempertahankan posisi setiap elemen sehingga data dapat diakses berdasarkan indeks, sedangkan struktur lainnya lebih menekankan pada keanggotaan elemen atau hubungan antara suatu kunci dengan nilai. Perbedaan cara pengorganisasian tersebut menentukan operasi yang dapat dilakukan terhadap data (Zhao et al., 2021). Sebagai contoh, *list* dan *tuple* memungkinkan elemen diakses berdasarkan posisi, sedangkan *dictionary* menggunakan *key* sebagai mekanisme utama untuk mengakses nilai.

Daftar Pustaka

Abdul Kadhar, K. M., & Anand, G. (2021). Basics of Python Programming. In K. M. Abdul Kadhar & G. Anand (Eds.), *Data Science with Raspberry Pi: Real-Time Applications Using a Localized Cloud* (pp. 13–47). Apress. https://doi.org/10.1007/978-1-4842-6825-4_2

Castro, O., Bruneau, P., Sottet, J. S., & Torregrossa, D. (2024). Landscape of High-Performance Python to Develop Data Science and *Machine learning* Applications. *ACM Computing Surveys*, *56*(3). https://doi.org/10.1145/3617588

Gong, Y., Liu, G., Xue, Y., Li, R., & Meng, L. (2023). A survey on dataset quality in *machine learning*. *Information and Software Technology*, *162*. https://doi.org/10.1016/j.infsof.2023.107268

Gupta, P., & Bagchi, A. (2024a). Introduction to NumPy. In P. Gupta & A. Bagchi (Eds.), *Essentials of Python for Artificial Intelligence and Machine learning* (pp. 127–159). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-43725-0_4

Gupta, P., & Bagchi, A. (2024b). Python Language Basics. In P. Gupta & A. Bagchi (Eds.), *Essentials of Python for Artificial Intelligence and Machine learning* (pp. 87–126). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-43725-0_3

Iqbal, M. A. (2025). Python Data Structures. In M. A. Iqbal (Ed.), *Python for Agriculturists* (pp. 137–181). Springer Nature Switzerland. https://doi.org/10.1007/978-3-032-01442-9_5

Khemani, B., Patil, S., Kotecha, K., & Tanwar, S. (2024). A review of graph neural networks: concepts, architectures, techniques, challenges, datasets, applications, and future directions. *Journal of Big Data*, *11*(1), 18. https://doi.org/10.1186/s40537-023-00876-4

Mafmudin, M., & Harnaningrum, L. N. (2025). Improving Memory Efficiency on Android: Leveraging Data Structures for Optimal Performance. *JIKO (Jurnal Informatika Dan Komputer)*, *9*(3), 496. https://doi.org/10.26798/jiko.v9i3.2131

Muddana, A. L., & Vinayakam, S. (2024a). Built-in Data Structure: Sets.

In A. L. Muddana & S. Vinayakam (Eds.), *Python for Data Science* (pp. 115–135). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-52473-8_6

Muddana, A. L., & Vinayakam, S. (2024b). *Python for data science* (1st ed.). Springer. https://doi.org/10.1007/978-3-031-52473-8

Shi, C. (2025). *Mastering Algorithms with Python* (1st ed.). Apress Berkeley, CA. https://doi.org/10.1007/979-8-8688-1799-1

Unpingco, J. (2021). *Python programming for data analysis* (1st ed.). Springer. https://doi.org/10.1007/978-3-030-68952-0

Wei, Y., Ben-David, N., Blelloch, G. E., Fatourou, P., Ruppert, E., & Sun, Y. (2021). Constant-time snapshots with applications to concurrent data structures. *Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPOPP*, 31–46. https://doi.org/10.1145/3437801.3441602

Zhao, W. X., Mu, S., Hou, Y., Lin, Z., Chen, Y., Pan, X., Li, K., Lu, Y., Wang, H., Tian, C., Min, Y., Feng, Z., Fan, X., Chen, X., Wang, P., Ji, W., Li, Y., Wang, X., & Wen, J. R. (2021). RecBole: Towards a Unified, Comprehensive and Efficient Framework for Recommendation Algorithms. *International Conference on Information and Knowledge Management, Proceedings*, 4653–4664. https://doi.org/10.1145/3459637.3482016

PROFIL PENULIS

Diki Arisandi, S.Kom., M.Kom., Ph.D.

Penulis memulai ketertarikan pada bidang ilmu komputer setelah menyelesaikan pendidikan menengah atas di SMA N 1 Bintan Timur, Kepulauan Riau. Saat ini penulis memiliki ketertarikan pada bidang *wireless network*, *cybersecurity*, *machine learning*, dan IT dalam bidang pendidikan. Penulis menyelesaikan pendidikan S1 di Universitas Abdurrab pada tahun 2010, dan memperoleh gelar Sarjana Komputer. Penulis melanjutkan studinya dan meraih gelar Magister Ilmu Komputer dari UPI "YPTK" Padang pada tahun 2013. Saat ini, penulis telah menyelesaikan studi S3 pada tahun 2026 di Multimedia University, Malaysia, dengan riset terkait *wireless network security*.

Penulis saat ini ber*home base* di Universitas Muhammadiyah Riau dan memiliki pengalaman mengajar di beberapa kampus yang ada di Pekanbaru. Penulis juga telah menerima berbagai hibah dari pendanaan internal maupun eksternal, seperti hibah dosen pemula, penelitian tingkat lanjut, penelitian fundamental, penelitian terapan, hibah pengabdian ipteks bagi masyarakat, dan pendayagunaan ruang publik. Selain itu, penulis juga memiliki beberapa karya buku ajar dan buku referensi terkait dengan bidang informatika yang berasal dari kegiatan Tridharma Perguruan Tinggi penulis.

Email Penulis: dikiarisandi@umri.ac.id

BAB 5
PEMROGRAMAN MODULAR

Fahmi, S.Kom., M.Kom.
Universitas Swadaya Gunung Jati

Pendahuluan

Dalam pengembangan perangkat lunak modern, dua paradigma pemrograman yang semakin populer dan relevan adalah pemrograman modular dan pemrograman fungsional. Keduanya menawarkan pendekatan berbeda untuk membangun aplikasi yang lebih terstruktur, dapat dipelihara, dan mudah diuji. Dengan meningkatnya kompleksitas sistem perangkat lunak, pengembang semakin memerlukan metode yang dapat mengelola kode secara efisien dan fleksibel (Teknologi Telekomunikasi, 2026).

Pemrograman modular merupakan pendekatan dalam pemrograman yang menekankan pembagian sebuah program ke dalam bagian-bagian kecil yang disebut modul. Setiap modul dirancang untuk menangani satu fungsi atau tugas tertentu sehingga keseluruhan program menjadi lebih terstruktur dan mudah dipahami. Dengan membagi program ke dalam modul-modul yang terpisah, pengembang dapat fokus pada satu bagian logika tanpa harus memahami seluruh isi program pada saat yang bersamaan.

Konsep pemrograman modular muncul sebagai solusi atas kompleksitas program yang semakin meningkat seiring berkembangnya kebutuhan sistem. Program yang ditulis dalam satu kesatuan besar tanpa pemisahan logika akan sulit dibaca, dipelihara, dan dikembangkan. Pemrograman modular membantu mengelola kerumitan dengan cara memecah program menjadi unit-unit kecil yang memiliki tanggung jawab jelas dan saling mendukung satu sama lain.

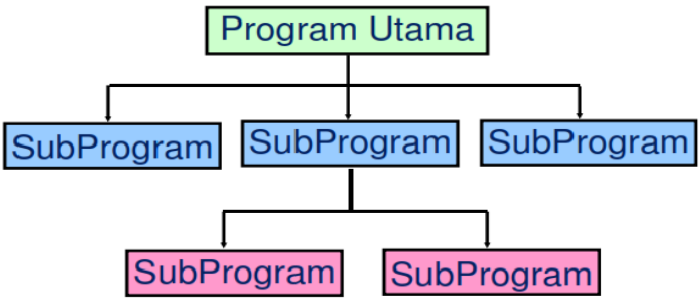
Dalam pemrograman modular, setiap modul dapat dianggap sebagai blok bangunan yang membentuk program. Modul-modul tersebut dapat berupa fungsi, prosedur, atau subprogram yang memiliki *input* dan *output* tertentu. Pembagian ini membuat alur program menjadi lebih rapi dan logis karena setiap modul hanya mengerjakan tugas yang telah ditentukan sebelumnya. Sub-sub program ini ada yang menamakannya subrutin, modul, prosedur, atau fungsi. Subprogram yang paling umum digunakan adalah prosedur dan fungsi. Ada perbedaan antara keduanya, namun pada praktiknya prosedur dapat dibuat fungsi dan fungsi dapat dibuat prosedur. Tidak

ada ikatan baku; ini lebih kepada ketaatan programmer itu sendiri terhadap kaidah-kaidah algoritma yang berlaku.

Konsep Pemrograman Modular

Jika kita memperhatikan sebuah mobil, maka mobil tersebut terdiri dari ban, kaca spion, pintu mobil, bagasi dan lain-lain. Saat terjadi kerusakan pada ban, kita dapat memperbaikinya atau menggantinya dengan yang baru. Analogi sederhana tersebut menjelaskan konsep modular dengan tepat, di mana sebuah modul merupakan sebuah komponen dari sebuah sistem yang lebih besar dan masing-masing bekerja secara independen (Ichsan & Lubis, 2019).

Adapun bentuk skema dari pemrograman modular sebagai berikut (Saragih & Nabila, 2019):



Gambar 5.1: Skema Pemrograman Modular

Sumber : Rasyid Al-Hadi Saragih

Dari skema pada Gambar 5.1, dapat kita tarik kesimpulan bahwa pemrograman modular mempunyai keuntungan, yaitu;

1. Masalah yang kompleks dapat dijadikan masalah-masalah yang lebih sederhana
2. Masalah yang kompleks dapat dibagi menjadi modul-modul yang lebih sederhana
3. Mencari kesalahan relatif lebih mudah, karena alur logika lebih jelas, kesalahan juga dapat dilokalisasi dalam satu modul
4. Modifikasi dapat dilakukan tanpa mengganggu program secara keseluruhan

Daftar Pustaka

Ichsan, M., & Lubis, K. A. (2019). Pemrograman Modular. In *Pendidikan Teknologi Informatika dan Komputermputer* (Vol. 1, Issue 1).

Madyamadja, E. D. (2026). *Pemrograman Modular (Modularisasi)*.

Manalu, I. P. (2023). *Mengenal Parameter pada Bidang Pemrograman*. Https://Www.Dicoding.Com/. https://www.dicoding.com/blog/mengenal-parameter-pada-bidang-pemrograman/

Munggaran, D. R., Almalik, A., & Fadhilah, Z. F. (2022). *Prosedur dan Fungsi Python* (Issue 1).

Rahman. (2024). *Modul Praktikum Pemrograman Terstruktur*.

Ramadhani, C. A., Miranda, G., & Nurhafizah. (2021). *Algoritma Pemrograman Modular* (Vol. 3, Issue 1).

Safriani, D., & Marnis, R. (2022). *Membangun Aplikasi Sederhana yang Menerapkan Prinsip -Prinsip Pemrograman Modula yang Baik dalam Masalah Nyata* (Issue 1).

Saragih, R. A.-H., & Nabila, R. (2019). *Pengertian Pemrograman Modular, Keuntungan Pemrograman Modular* (Issues 1–11).

Teknologi Telekomunikasi. (2026). *Pemrograman Modular dan Fungsional*. Https://Dte.Telkomuniversity.Ac.Id/. https://dte.telkomuniversity.ac.id/en/pemrograman-modular-dan-fungsional/

PROFIL PENULIS

Fahmi, S.Kom., M.Kom.

Penulis terlahir dan dibesarkan di Cirebon. Mengenyam pendidikan dari SD hingga SMA di kota kelahiran yang dikenal sebagai Kota Udang. Sejak SMA sudah tertarik dengan bidang ilmu komputer dan teknologi, hingga mengantarkan penulis untuk menempuh pendidikan ke Perguruan Tinggi dan berhasil menyelesaikan studi S1 di program studi Teknik Informatika Sekolah Tinggi Ilmu Komputer (STIKOM) Poltek Cirebon pada tahun 2008. Berselang di tahun 2015, penulis menyelesaikan studi S2 di program Magister Sistem Informasi Program Pascasarjana Universitas Komputer Indonesia (UNIKOM) Bandung.

Penulis memiliki keahlian di bidang Web Technology dan Rekayasa Perangkat Lunak. Dan untuk mewujudkan karier sebagai dosen profesional, penulis pun aktif sebagai peneliti di bidang keahliannya tersebut. Selain sebagai peneliti, penulis juga aktif melakukan pengabdian kepada masyarakat dan menulis karya ilmiah untuk publikasi jurnal sebagai bentuk implementasi tri dharma sebagai seorang dosen.

Saat ini menjadi dosen di Program Studi Informatika Fakultas Teknik Universitas Swadaya Gunung Jati yang terletak di Kota Cirebon. Adapun mata kuliah yang pernah diampu antara lain Logika Informatika, Metode Numerik, Aljabar Linear, Pemrograman Berorientasi Objek, Kalkulus, dan Database Administrator.

Email Penulis: fahmi@ugj.ac.id

BAB 6
KONSEP DASAR OBJECT-ORIENTED PROGRAMMING (OOP)

Tugiman, S.Kom., M.Kom., IPM.
Sekolah Tinggi Teknologi NIIT – I-Tech

Teknologi informasi yang semakin berkembang belakangan ini dengan semakin pesat telah mendorong kebutuhan terhadap perangkat lunak yang tidak hanya mampu menjalankan fungsi tertentu, tetapi juga mudah dikembangkan, dipelihara, dan disesuaikan dengan perubahan kebutuhan. Awal perkembangan pemrograman menggunakan model terstruktur atau prosedural. Pemrograman model ini banyak digunakan dengan menempatkan algoritma, prosedur, dan fungsi sebagai pusat pengorganisasian program. Pendekatan tersebut cukup efektif untuk menyelesaikan persoalan dengan tingkat kompleksitas yang terbatas.

Namun, ketika ukuran perangkat lunak semakin besar dan melibatkan banyak komponen yang saling berinteraksi, pengelolaan data dan fungsi secara terpisah dapat menimbulkan persoalan dalam pemeliharaan, pengembangan, serta penggunaan kembali kode. Kondisi tersebut mendorong berkembangnya paradigma *Object-Oriented Programming* (OOP), yaitu pendekatan pemrograman yang mengorganisasikan perangkat lunak berdasarkan objek yang memiliki data dan perilaku. Dalam paradigma ini, konsep seperti *class*, *object*, *attribute*, *method*, *encapsulation*, *inheritance*, *abstraction*, dan *polymorphism* menjadi bagian penting dalam membangun struktur perangkat lunak yang lebih terorganisasi. Object-Oriented Programming, atau disingkat OOP, adalah salah satu paradigma pemrograman paling populer dan banyak digunakan saat ini. Dari aplikasi *mobile*, *web*, hingga sistem *enterprise* kompleks, OOP menjadi fondasi penting dalam pengembangan perangkat lunak modern

Pendahuluan

Perkembangan perangkat lunak dari waktu ke waktu menunjukkan adanya perubahan cara pandang dalam merancang dan membangun program. Pada tahap awal, pemrograman banyak dikembangkan dengan pendekatan prosedural yang menempatkan urutan instruksi, fungsi, dan prosedur sebagai bagian utama dalam penyelesaian masalah. Pendekatan tersebut sesuai untuk program dengan lingkup dan kompleksitas yang relatif terbatas. Namun, meningkatnya kebutuhan terhadap aplikasi yang memiliki banyak data, fungsi, pengguna, serta hubungan antarkomponen menyebabkan proses

pengembangan perangkat lunak menjadi semakin kompleks. Kondisi tersebut mendorong munculnya berbagai paradigma pemrograman yang berusaha memberikan cara yang lebih sistematis dalam mengorganisasikan program.

Salah satu paradigma yang berkembang dan banyak digunakan adalah *Object-Oriented Programming* (OOP), yaitu pendekatan yang memandang perangkat lunak sebagai kumpulan objek yang memiliki data, perilaku, dan hubungan dengan objek lainnya. *Object-Oriented Programming* (OOP) adalah pendekatan yang memandang perangkat lunak sebagai kumpulan objek yang memiliki data, perilaku, dan hubungan dengan objek lainnya. Dalam pendekatan ini, permasalahan tidak hanya dipandang sebagai rangkaian langkah penyelesaian, tetapi juga sebagai kumpulan entitas yang perlu direpresentasikan secara terstruktur di dalam perangkat lunak (Indahyanti & Astutik, 2025), (Hadiprakoso, 2021).

Tujuan dari penggunaan OOP adalah untuk mempermudah pekerjaan programmer ketika melakukan pengembangan program dengan mengikuti model yang telah ada dalam kehidupan sehari-hari, menjadikan pemrograman lebih fleksibel, dan dapat digunakan secara luas dalam skala besar. Dalam sebuah program, sebuah objek yang besar dibentuk dari gabungan beberapa objek yang lebih kecil, di mana objek-objek tersebut saling berkomunikasi dan bertukar informasi dengan objek yang lain untuk mencapai hasil akhir.

Perubahan paradigma tersebut berkaitan erat dengan perkembangan kebutuhan terhadap perangkat lunak yang semakin besar dan dinamis. Pada pemrograman prosedural, data dan fungsi sering kali dikelola sebagai bagian yang relatif terpisah, sehingga ketika aplikasi berkembang, keterkaitan antarbagian program dapat menjadi semakin sulit untuk dikelola. Perubahan terhadap struktur data tertentu juga dapat menimbulkan dampak terhadap sejumlah fungsi yang menggunakan data tersebut. Kondisi seperti ini dapat meningkatkan kompleksitas pemeliharaan dan menyulitkan pengembangan fitur baru.

OOP menawarkan pendekatan yang menggabungkan data dan perilaku yang berkaitan ke dalam suatu objek, sehingga struktur program dapat disusun berdasarkan tanggung jawab masing-masing

Daftar Pustaka

Adiputra, F. (2022). *Pemrograman Berorientasi Objek dengan Java*. Media Nusa Kreatif.

Anif, M., Samsinar, & Anggoro, D. (2024). *Pemrograman Berorientasi Objek*. Deepublish.

Asnawi, N. (2023). *Teori dan Praktik Object Oriented Programming Menggunakan Java*. UNIPMA Press.

Fikri, M. A., Kurniasari, N., Dhaniswara, E., Tampati, N., Swarga, L. T., & Marhaendra, A. (2025). *Pemrograman Berbasis Objek Dasar*. Bima Utama Press.

Hadiprakoso, R. B. (2021). *Pemrograman Berorientasi Objek: Teori dan Implementasi dengan Java*. RBH.

Indahyanti, U., & Astutik, I. R. I. (2025). *Buku Ajar Pemrograman Berorientasi Objek (PBO)*. UMSIDA Press. https://doi.org/10.21070/2025/978-623-464-130-1

Jayadi, P. (2024). *Pemrograman Berorientasi Obyek: Konsep dan Implementasi*. Deepublish Digital.

Kementerian Pendidikan Dasar dan Menengah Republik Indonesia. (2025). *Panduan Mata Pelajaran Koding dan Kecerdasan Artifisial*. https://repositori.kemendikdasmen.go.id/33639/1/33. Final Panduan Mata Pelajaran Panduan Mata Pelajaran Koding dan Kecerdasan Artifisial_12_Sep_2025_revisi 3.pdf

Nofriadi, Risnawati, Lubis, A. P., & Dalimunthe, R. A. (2024). *Cara Pintas Menguasai Pemrograman Berorientasi Objek*. Deepublish.

Nugroho, A. Y. (2025). *Pemrograman Berorientasi Objek*. Bravo Press Indonesia.

Object Management Group. (2012). *OMG Unified Modeling Language (OMG UML), Version 2.4.1*. https://www.omg.org/spec/UML/2.4.1/

Pusat Asesmen Pendidikan, Kementerian Pendidikan Dasar dan Menengah Republik Indonesia. (2025). *Pengembangan Perangkat Lunak dan Gim: Pemrograman Berorientasi Objek*. https://pusmendik.kemdikbud.go.id/tka/tka/view/mata-pelajaran-pilihan/sma/pengembangan-perangkat-lunak-dan-gim-

smk-mak

Pusat Asesmen Pendidikan. (2026). *Pengembangan Perangkat Lunak dan Gim: Pemrograman Berorientasi Objek*. https://pusmendik.kemdikbud.go.id/tka/tka/view/mata-pelajaran-pilihan/sma/pengembangan-perangkat-lunak-dan-gim-smk-mak

Putra, Y. W. S., Handayani, R. D., Astuti, D., Arifah, F. N., Nasution, E., Sutjiningtyas, S., Meidelfi, D., Kanafi, & Sukma, F. (2024). *Pemrograman Berorientasi Objek Dengan Java dan NetBeans IDE*. Tohar Media.

Sasono, I., & Nugroho, A. (2024). *Pemrograman Berorientasi Objek Menggunakan Python*. Pahin Media Kreasi.

Setiawan, A., Ambiyar, Mukhaiyar, R., Yanto, B., & Fimawahib, L. (2024). *Struktur Data Pendekatan Pemrograman Berorientasi Objek*. Deepublish.

Sujjada, A., & Somantri. (2024). *Buku Ajar Pemrograman Berorientasi Objek*. Kaizen Media Publishing.

Wicaksono, F. (2023). *Buku Ajar Pemrograman Berorientasi Objek Lanjut*. Deepublish

PROFIL PENULIS

Tugiman, S.Kom., M.Kom., IPM.

Penulis lahir di Boyolali, 15 September 1968. Lulus Diploma I dari Institut Manajemen Komputer Indonesia (IMKI) Surakarta tahun 1991. Menamatkan Sarjana Jurusan Sistem Informasi di Universitas Budi Luhur tahun 2014. Kemudian melanjutkan studi Program Pascasarjana Jurusan Sistem Informasi di Universitas Budi Luhur dan selesai pada tahun 2016. Sebagai Dekan Fakultas Sosial dan Teknologi Universitas Medika Suherman 2022-2025. Saat ini, sebagai Kaprodi di Sekolah Tinggi Teknologi NIIT – I-Tech Jakarta Selatan, mengajar di Universitas Buddhi Dharma Tangerang, dan sebagai konsultan di sebuah Rumah Sakit. Organisasi yang diikuti saat ini adalah sebagai Pengurus Asosiasi Perguruan Tinggi Manajemen Ritel Indonesia (APTMRI), sebagai pengurus DPW Ikatan Ahli Informatika Indonesia (IAII) Banten, sebagai pengurus pusat Badan Kejuruan Informatika (BKI) Persatuan Insinyur Indonesia (PII), sebagai anggota Gugus Tugas Penyusunan Standar Layanan Insinyur-Kementrian Teknis (Kementrian Kesehatan), sebagai anggota APTIKOM, sebagai anggota Asosiasi Dosen Indonesia (ADI). Adapun mata kuliah yang pernah diampu adalah Rekayasa Perangkat Lunak, Manajemen Proyek, Analisa dan Perancangan Sistem Informasi, Audit Sistem Informasi, E-Bisnis, E-Commerce, IT Budgeting, Manajemen Operasi, Strategi Pemasaran, Testing dan Implemetasi, dan Manajemen Sumber Daya Manusia. Selain itu juga pernah mengerjakan beberapa aplikasi yang dipakai di UMKM dan Rumah Sakit. Email Penulis: tugiman0311@gmail.com

BAB 7
PRINSIP *OBJECT ORIENTED PROGRAMMING* (OOP) LANJUTAN

Ahmad Budi Trisnawan, S.T., M.Kom.
Universitas Mahakarya Asia, Jakarta

Pendahuluan

Perkembangan teknologi perangkat lunak yang semakin kompleks menuntut *programmer* tidak hanya mampu menulis kode yang dapat menjalankan suatu fungsi, tetapi juga mampu merancang program yang terstruktur, *modular,* dapat digunakan kembali, mudah dipelihara, dan dapat dikembangkan. Salah satu pendekatan pemrograman yang banyak digunakan untuk memenuhi kebutuhan tersebut adalah *Object-Oriented Programming* (OOP) atau pemrograman berorientasi objek (Rahman et al., 2023).

Python merupakan salah satu bahasa pemrograman modern yang mendukung paradigma OOP secara kuat. Hampir seluruh komponen dalam *Python* dapat dipandang sebagai objek, sehingga pemahaman mengenai *class, object, attribute, method,* dan hubungan antarkelas menjadi fondasi penting dalam membangun aplikasi *Python* (Raharjo, 2022). Konsep OOP juga banyak digunakan dalam berbagai ekosistem teknologi, khususnya *Data Science, Artificial Intelligence,* dan *Machine Learning.* Berbagai *library* populer, seperti *scikit-learn*, *PyTorch*, dan *TensorFlow* menggunakan pendekatan berbasis objek dalam membangun model, mengelola data, melakukan transformasi, serta menjalankan proses komputasi (Suharto, 2023).

Pada pembahasan OOP tingkat dasar, *programmer* umumnya telah mengenal pembuatan *class* dan *object* serta memahami konsep *encapsulation, inheritance, polymorphism,* dan *abstraction* (Zein & Tristianto, 2025). Namun, ketika program mulai berkembang menjadi sistem yang lebih besar, penerapan konsep tersebut membutuhkan pemahaman yang lebih mendalam. *Programmer* perlu mengetahui bagaimana membangun hierarki *class,* melakukan pewarisan secara tepat, mengimplementasikan *polymorphism,* menyembunyikan kompleksitas melalui *abstraction,* serta menjaga integritas data melalui *encapsulation* (Mulyono & Saleh, 2025).

OOP lanjutan juga tidak hanya berkaitan dengan bagaimana sebuah *class* dibuat, tetapi lebih jauh membahas bagaimana *class* dirancang dan berinteraksi dengan komponen lainnya (Pamungkas et al., 2020). Kesalahan dalam merancang hubungan antarkelas dapat menyebabkan kode menjadi sulit dipelihara, memiliki ketergantungan

yang tinggi, dan sulit dikembangkan (Rahman et al., 2023). Oleh karena itu, pembahasan OOP modern perlu diarahkan pada prinsip desain yang menghasilkan kode yang lebih *modular* dan memiliki tanggung jawab yang jelas.

Dalam konteks *machine learning,* penerapan OOP menjadi semakin penting karena proses pengembangan model biasanya terdiri atas beberapa tahapan yang saling berhubungan, mulai dari pengambilan data, *preprocessing, feature engineering,* pelatihan model, evaluasi, hingga prediksi (Raharjo, 2022). Setiap tahapan tersebut dapat direpresentasikan sebagai objek atau komponen yang memiliki tanggung jawab tertentu. Pendekatan seperti ini memungkinkan sebuah sistem *machine learning* dibangun secara *modular,* sehingga algoritma atau komponen tertentu dapat diganti tanpa harus mengubah keseluruhan sistem (Suharto, 2023).

Bab ini membahas prinsip-prinsip *Object-Oriented Programming* (OOP) lanjutan menggunakan *Python* melalui tiga fokus utama. Pertama, pembahasan mengenai pewarisan dan hierarki kelas lanjutan, termasuk *inheritance, method overriding,* penggunaan super(), dan *multiple inheritance.* Kedua, pembahasan mengenai *polymorphism, abstraction,* dan *interface* sebagai dasar untuk membangun program yang fleksibel dan memiliki kontrak perilaku yang jelas. Ketiga, pembahasan mengenai *encapsulation* dan desain OOP untuk sistem modern, termasuk penggunaan *property,* prinsip *Single Responsibility,* serta *composition* sebagai pendekatan alternatif terhadap *inheritance* (Zein & Tristianto, 2025).

Setelah mempelajari bab ini, pembaca diharapkan tidak hanya mampu membuat *class* dan *object* dalam *Python,* tetapi juga mampu merancang struktur program berbasis OOP yang *modular, reusable, maintainable,* dan *scalable* (Mulyono & Saleh, 2025). Pemahaman tersebut menjadi salah satu fondasi penting sebelum pembaca memasuki pembahasan yang lebih kompleks mengenai struktur data, algoritma, *Data Science,* dan implementasi *machine learning* menggunakan *Python.*

Daftar Pustaka

Alimuddin, A. A. P. (2025). *Pemrograman Berorientasi Objek: Konsep, Praktik, dan Aplikasi dalam Bahasa Java* (1st ed.). Eureka Media Aksara.

Anif, M., Samsinar, S., & Anggoro, D. (2024). *Pemrograman Berorientasi Objek*. Deepublish Digital.

Camilo, T. M., Virginia, G., Susanto, B., & Proboyekti, U. (2021). Pemodelan Representasi Pengetahuan Berbasis OWL untuk Objek Arsitektur Candi di Indonesia. *Jurnal Terapan Teknologi Informasi*, *4*(1), 13–21. https://doi.org/10.21460/jutei.2020.41.190

Hidayat, R., Trisnawan, A. B., Slam, B. E., Subiksa, G. B., Herikson, R., Irawati, N., Ichwani, A., Betty Yel, M., Romodon, D., Riza, F., Prayudha, I. P. A., & Anugrah, R. (2026). *Object-Oriented Programming* (1st ed.). PT. Faaslib Serambi Media. https://faaslibsmedia.com/

Ida, I., Faisal, M., Musdalifa, S., Rosnani, R., Nasi, N., & Moeis, D. (2024). *Buku Ajar Pemrograman Berorientasi Objek* (1st ed.). PT. Inovasi Pratama Internasional.

Indahyanti, U., & Astutik, I. R. I. (2025). *Buku Ajar Pemrograman Berorientasi Objek (PBO)* (1st ed.). UMSIDA Press.

Mulyono, S., & Saleh, Y. K. P. (2025). *Python untuk Data Science: Analisis Data, Visualisasi, dan Pembelajaran Mesin* (1st ed.). PT. Media Penerbit Indonesia.

Nugroho, A. Y. (2025). *Pemrograman Beorientasi Objek* (1st ed.). CV. Bravo Press Indonesia. www.bravopress.id

Nurhadi, N. (2023). *Buku Ajar: Pemrograman Berorientasi Objek Menggunakan Visual Basic* (1st ed.). Hadla Media Informasi. www.media.hadlacorp.com

Pamungkas, F. S., Prasetya, B. D., & Kharisudin, I. (2020). Perbandingan Metode Klasifikasi Supervised Learning pada Data Bank Customers Menggunakan Python. *PRISMA, Prosiding Seminar Nasional Matematika*, *3*, 689–694. https://journal.unnes.ac.id/sju/index.php/prisma/

Raharjo, B. (2022). *Deep Learning dengan Python*. Yayasan Prima Agus Teknik.

Rahman, S., Sembiring, A., Siregar, D., Khair, H., Prahmana, G. I., Puspadini, R., & Zen, M. (2023). *Python: Dasar dan Pemrograman Berorientasi Objek* (1st ed.). Tahta Media Group.

Santoso, J. T. (2023). *Pemrograman Berorientasi Objek dengan Java BlueJ*. Yayasan Prima Agus Teknik.

Suharto, A. (2023). *Fundamental Data Science dengan Library Pandas Python* (1st ed.). CV. Eureka Media Aksara.

Zein, A., & Tristianto, C. (2025). *Algoritma dan Struktur Data Menggunakan Pemrograman Python* (1st ed.). CV. Eureka Media Aksara.

PROFIL PENULIS

Ahmad Budi Trisnawan, S.T., M.Kom.

Ketertarikan penulis terhadap ilmu komputer dimulai pada tahun 2010 silam. Hal tersebut membuat penulis memilih untuk masuk ke Sekolah Menengah Kejuruan di SMK Negeri 2 Kota Tangerang dan berhasil lulus pada tahun 2010. Penulis kemudian melanjutkan pendidikan ke Perguruan Tinggi dan berhasil menyelesaikan studi S1 pada prodi Teknik Informatika Universitas Satya Negara Indonesia pada tahun 2014. Lima tahun kemudian, penulis menyelesaikan studi S2 pada Prodi Ilmu Komputer Program Pasca Sarjana Universitas Budi Luhur.

Penulis memiliki dua (2) buah hati yakni Ardanu Fatih Trisnawan dan Bahira Freya Trisnawan dari pasangan Budi Lestiarini, S.E. Penulis memiliki kepakaran dibidang Sistem Informasi, Data Mining, dan Big Data. Dan untuk mewujudkan karir sebagai dosen profesional, penulis pun aktif sebagai peneliti dibidang kepakarannya tersebut. Selain peneliti, penulis juga aktif menulis buku dengan harapan dapat memberikan kontribusi positif bagi bangsa dan negara yang sangat tercinta ini. Email Penulis: abudit75@gmail.com

BAB 8
DESAIN POLA DAN METODE KHUSUS

Eza Budi Perkasa, M.Kom.
Institut Sains dan Bisnis Atma Luhur

Desain Pola

Desain pola merupakan solusi umum untuk permasalahan yang biasanya terjadi dalam desain dan implementasi perangkat lunak. Desain tersebut dapat dilakukan berulang kali sesuai dengan kebutuhan perangkat lunak bersangkutan. Tujuan yang hendak dicapai dari desain pola adalah mempercepat proses pengembangan dengan menyediakan paradigma pengembangan atau desain yang telah teruji dan terbukti sebelumnya (GeeksforGeeks, t.t.).

Hal yang perlu diperhatikan dalam penggunaan desain pola adalah sebagai berikut.

1. Desain pola tidak tergantung pada bahasa pemrograman yang digunakan untuk memecahkan masalah umum.
2. Desain pola tidak wajib diterapkan dalam proyek karena tidak dimaksudkan untuk pengembangan proyek. Desain pola hanya digunakan sesuai kebutuhan untuk mencari solusi atas masalah umum yang terjadi di proyek.
3. Desain pola tidak perlu dihafalkan; hanya perlu dicoba untuk memahami tujuan dari setiap pola yang ditawarkan.

Karakteristik Desain Pola

Desain pola memiliki karakteristik:

1. Mengidentifikasi masalah desain umum dan memberikan solusi yang terdefinisi dengan baik.
2. Meningkatkan penggunaan ulang kode dengan merangkum praktik yang berhasil sehingga dapat menghemat waktu dan tenaga.
3. Mengabstraksi detail implementasi tertentu dan berfokus pada konsep desain tingkat tinggi.
4. Membentuk *vocabulary* umum dan himpunan *term* yang dapat digunakan oleh pengembang perangkat lunak.
5. Didasarkan pada pengalaman kolektif komunitas pengembang perangkat lunak.

Berdasarkan karakteristik tersebut, desain pola memiliki kelebihan sebagai berikut.

1. Menekankan penciptaan solusi umum untuk berbagai proyek.
2. Menyediakan pendekatan terstruktur terhadap desain.

3. Menggunakan *vocabulary* dan abstraksi umum.
4. Berkontribusi pada kode basis yang modular dan terorganisiasi.
5. Menawarkan solusi siap pakai untuk permasalahan umum.
6. Berfungsi sebagai alat pemecahan masalah dalam pengembangan perangkat lunak.

Di samping itu, desain pola juga memiliki sejumlah kekurangan yang perlu dipertimbangkan sebagai berikut.

1. Memerlukan pengalaman dan pengetahuan untuk memahami dan menerapkan pola, mengingat pola yang akan digunakan adalah penggunaan ulang dari sebelumnya.
2. Tanpa kebutuhan atau pemahaman yang jelas dari pengembang, penggunaan desain pola berisiko akan mengakibatkan rekayasa yang berlebihan.
3. Penerapan desain pola berpotensi menghasilkan kode yang kaku.
4. Tidak semua pola dapat diterapkan, atau dengan kata lain, harus dipilih pola yang sekiranya tepat untuk permasalahan yang dihadapi.
5. Perangkat lunak yang menggunakan desain pola dapat menimbulkan tantangan dalam proses pemeliharaan.

Jenis Pola

Desain pola dapat dikelompokkan menjadi beberapa jenis sesuai dengan pendekatan yang digunakan. Ringkasan kelompok desain pola dapat dilihat pada Gambar 8.1.

Daftar Pustaka

GeeksforGeeks. (t.t.). *Design Patterns Tutorial*. Noida: Sanchaya Education Private Limited. Diakses 3 September 2026, dari https://www.geeksforgeeks.org/system-design/software-design-patterns/

Python Software Foundation. (2001). *The Python Language Reference*. Diakses 3 September 2026, dari https://docs.python.org/3/reference/

PROFIL PENULIS

Eza Budi Perkasa, M.Kom

Penulis merupakan salah satu dosen di Institut Sains dan Bisnis Atma Luhur, Program Studi Teknik Informatika sejak tahun 2016. Penulis mulai tertarik pada dunia komputer sejak SD dan mulai mempelajarinya lebih dalam sejak SMA melalui pelatihan Olimpiade Sains Nasional bidang Informatika pada tahun 2008. Ketertarikan tersebut membuat penulis melanjutkan pendidikannya ke Sekolah Tinggi Manajemen Informatika dan Komputer Atma Luhur untuk jenjang Strata 1 program studi Teknik Informatika pada tahun 2010. Saat itu juga, penulis berhak memperoleh beasiswa untuk melanjutkan ke Universitas Budi Luhur jenjang Strata 2 program studi Magister Ilmu Komputer pada tahun 2014.

Penulis memiliki kepakaran di bidang kecerdasan tiruan dan algoritma. Penelitian yang dilakukan didanai oleh internal perguruan tinggi dan juga Kementerian Pendidikan Tinggi, Sains, dan Teknologi. Penulis juga memiliki prestasi lainnya berupa peraih medali emas di Olimpiade Nasional 2023 jenjang Umum bidang Matematika, medali perunggu di Olimpiade Rihand Creative Guru 2023 bidang TIK, finalis di Olimpiade Rihand Creative Guru 2023 bidang Matematika, medali perak di BNSO 2024 jenjang Guru/Pascasarjana/Umum bidang Bahasa Inggris, serta *absolute winner* di BNSO 2024 jenjang Guru/Pascasarjana/Umum bidang Astronomi dan Matematika. Saat ini, penulis juga menduduki jabatan sebagai Bendahara di Asosiasi Pendidikan Tinggi Informatika dan Komputer (APTIKOM) Provinsi Kepulauan Bangka Belitung periode 2026-2030.

Email Penulis: ezabudiperkasa@atmaluhur.ac.id

BAB 9
INPUT/OUTPUT DAN SERIALISASI DATA

Saryani, S.Kom., M.TI.
Universitas Tangerang Raya

Pendahuluan *Input*/*Output* (I/O)

Input/*Output* (I/O) merupakan bagian fundamental dalam pemrograman karena program tidak hanya melakukan perhitungan di dalam memori, tetapi juga perlu menerima data dari pengguna, membaca data dari berkas, mengirimkan keluaran ke layar, serta menyimpan hasil pengolahan agar dapat digunakan kembali. Dalam aplikasi modern, data juga perlu dipertukarkan antarprogram dan antarsistem dengan format yang terstruktur. Kebutuhan tersebut menjadikan serialisasi dan deserialisasi sebagai konsep penting dalam pengembangan perangkat lunak.

Bab ini membahas konsep I/O, *input* dari pengguna, *output*, pengelolaan berkas, format CSV dan JSON, konsep serialisasi dan deserialisasi, pemilihan format data, aspek keamanan, serta contoh implementasi menggunakan Python. Python digunakan sebagai bahasa contoh karena sintaksnya ringkas dan mendukung berbagai format data melalui pustaka bawaan maupun pustaka pihak ketiga.

Konsep Dasar *Input*/*Output*

Input adalah data yang masuk ke dalam program untuk diproses, sedangkan *output* adalah informasi yang dihasilkan program setelah melakukan proses tertentu. Sumber *input* dapat berupa *keyboard*, berkas, basis data, sensor, jaringan, atau layanan aplikasi. *Output* dapat ditampilkan pada terminal, ditulis ke berkas, dikirim melalui jaringan, atau diteruskan ke komponen aplikasi lain.

Secara konseptual, aliran data dalam program dapat digambarkan sebagai *input* → proses → *output*. Pada aplikasi yang lebih kompleks, aliran tersebut dapat berlangsung berulang dan melibatkan penyimpanan permanen. Karena itu, program perlu menangani kondisi seperti data kosong, tipe data yang tidak sesuai, berkas yang tidak ditemukan, izin akses, serta kesalahan format.

Karakteristik *Input* dan *Output*:

1. *Input* harus divalidasi agar data yang diterima sesuai dengan kebutuhan program.
2. *Output* harus disajikan dalam bentuk yang mudah dipahami atau mudah diproses kembali.

3. Operasi I/O umumnya melibatkan media eksternal sehingga dapat lebih lambat daripada operasi di memori.
4. Operasi I/O dapat menghasilkan kesalahan, sehingga program perlu menyediakan mekanisme penanganan *exception*.

Input dari Pengguna

Input sederhana pada aplikasi berbasis terminal dapat diperoleh menggunakan fungsi *input*(). Nilai yang dikembalikan fungsi tersebut berupa *string,* sehingga konversi diperlukan apabila program membutuhkan bilangan atau tipe data lain.

```
nama = input("Masukkan nama: ")
umur = int(input("Masukkan umur: "))

print("Nama:", nama)
print("Umur:", umur)
```

Pada contoh tersebut, *input* umur dikonversi menjadi integer menggunakan int(). Konversi dapat menimbulkan *Value*Error apabila pengguna memasukkan teks yang bukan bilangan. Oleh sebab itu, *input* pada aplikasi nyata sebaiknya divalidasi.

```
while True:
    try:
        umur = int(input("Masukkan umur: "))
        if umur < 0:
            raise ValueError("Umur tidak boleh
negatif.")
        break
    except ValueError as e:
        print("Input tidak valid:", e)
```

Validasi bukan hanya persoalan tipe data. Program juga perlu memeriksa rentang nilai, panjang teks, format tanggal, pilihan yang tersedia, serta aturan bisnis. Validasi yang baik mengurangi risiko kesalahan proses dan membuat pengalaman pengguna lebih baik.

Output Program

Output merupakan informasi yang dihasilkan program. Pada Python, fungsi print() digunakan untuk menampilkan informasi ke standar *output*. Pengembang dapat mengatur format keluaran agar konsisten dan mudah dibaca.

Daftar Pustaka

Python Software Foundation. (2026). Built-in types. Python 3 documentation.

Python Software Foundation. (2026). *Input* and *output*. Python 3 documentation.

Python Software Foundation. (2026). json — JSON encoder and decoder. Python 3 documentation.

Python Software Foundation. (2026). csv — CSV File Reading and Writing. Python 3 documentation.

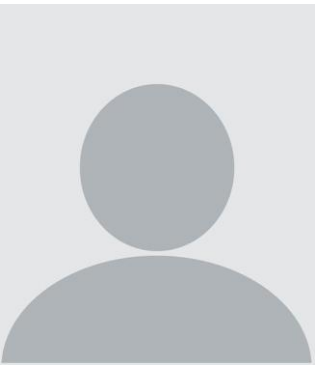
Python Software Foundation. (2026). pickle — Python object serialization. Python 3 documentation.

Ramalho, L. (2022). Fluent Python (2nd ed.). O'Reilly Media.

Sweigart, A. (2023). Automate the boring stuff with Python (3rd ed.). No Starch Press.

Van Rossum, G., & Drake, F. L. (Eds.). (2009). Python 3 reference manual. CreateSpace.

PROFIL PENULIS



Hj. Saryani, S.Kom., M.TI.

Penulis merupakan dosen tetap dan Ketua Program Studi Teknologi Informasi pada Universitas Tangerang Raya. Penulis memiliki latar belakang pendidikan di bidang teknologi informasi dan memiliki minat akademik pada analisis dan perancangan sistem informasi, metode penelitian, public speaking, dan teknologi platform. Dalam kegiatan akademik, penulis aktif melaksanakan pendidikan, penelitian, dan pengabdian kepada masyarakat, serta terlibat dalam pengembangan kurikulum, implementasi pembelajaran berbasis capaian pembelajaran, dan penguatan kompetensi mahasiswa di bidang teknologi informasi. Penulis juga memiliki pengalaman dalam penyusunan RPS, pengembangan materi pembelajaran, pendampingan tugas akhir, serta kegiatan penelitian dan publikasi ilmiah. Ketertarikan pada perkembangan teknologi, khususnya pemanfaatan sistem informasi dan platform digital, mendorong penulis untuk terus mengembangkan bahan ajar yang kontekstual dan mudah diterapkan. Melalui buku ini, penulis berharap pembaca dapat memahami konsep pemrograman secara sistematis sekaligus mampu menerapkan teknik pengolahan *input*/*output* dan serialisasi data dalam pengembangan aplikasi.

Email Penulis: Saryani@untara.ac.id

BAB 10
PENGENALAN NUMPY

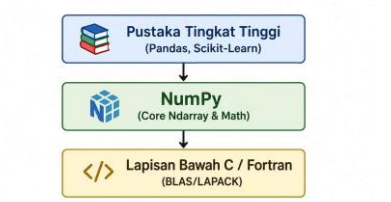
Ir. Indo Intan, S.T., M.T.
Universitas Dipa Makassar

Pengenalan Numpy dan Ekosistem Utama Saintek

1. Latar Belakang dan Sejarah Singkat Numpy

Bahasa pemrograman Python telah berkembang dari sekadar bahasa skrip umum menjadi standar *de facto* dalam ekosistem *data science*, *machine learning*, dan komputasi numerik modern (McKinney, 2022; Virtanen et al., 2020). Kendati demikian, Python murni memiliki keterbatasan bawaan (*inherent limitation*) terkait kecepatan eksekusi dan efisiensi memori karena sifatnya sebagai bahasa terinterpretasi (*interpreted language*) dengan mekanisme *Dynamic Typing* (VanderPlas, 2022).

Untuk mengatasi kendala tersebut, Travis Oliphant pada tahun 2005 menggabungkan peninggalan pustaka "Numeric" dan "Numarray" menjadi satu fondasi tunggal yang dinamakan "NumPy" (*Numerical Python*) (Harris et al., 2020). Kehadiran NumPy merevolusi paradigma pemrograman array (*array programming paradigm*) dengan menyediakan struktur data homogen berdimensi banyak yang dieksekusi di tingkat bahasa C (Harris et al., 2020). Saat ini, NumPy menjadi *core dependency* bagi hampir seluruh ekosistem saintifik Python, seperti Pandas, SciPy, Matplotlib, hingga Scikit-Learn (Virtanen et al., 2020; McKinney, 2022).



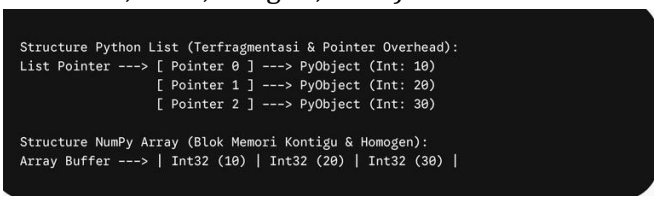
Gambar 10.1: Level Numpy pada Struktur Data
Sumber: Diolah penulis

2. Mengapa Numpy ? Perbandingan Performa: Python List Vs. Numpy Array

Perbedaan mendasar antara struktur data bawaan Python (list) dan array NumPy (ndarray) terletak pada alokasi memori dan mekanisme pemrosesan data (Harris et al., 2020; VanderPlas, 2022).

a. Struktur Memori Terfragmentasi vs. Kontigu
   1) Python List: merupakan array dari penunjuk (*array of pointers*). Setiap elemen dalam *list* menyimpan *pointer* yang merujuk ke objek Python lain yang tersebar secara acak di memori *heap* (VanderPlas, 2022). Hal ini menyebabkan *overhead* memori yang besar dan memicu fenomena *cache miss* pada CPU (Harris et al., 2020).
   2) NumPy Array (ndarray): menyimpan data dalam satu blok memori yang "kontigu" (*contiguous memory block*) dengan tipe data yang konstan/homogen (Harris et al., 2020). Karakteristik ini memungkinkan CPU mengakses data secara berurutan dengan efisiensi tinggi, memanfaatkan *CPU L1/L2 cache* (Harris et al., 2020; Rougier, 2021).

b. Penanganan Tipe Data
   1) Python List: Bersifat heterogen (*Dynamic Typing*). Setiap elemen adalah objek Python lengkap yang membawa metadata (*reference count*, tipe objek, ukuran) sehingga memerlukan proses *type checking* pada setiap iterasi (VanderPlas, 2022).
   2) NumPy Array: Bersifat homogen (*Static Typing* di tingkat C). Seluruh elemen memiliki ukuran bit dan tipe data yang persis sama, menghapuskan kebutuhan *type checking runtime* (Harris et al., 2020; Rougier, 2021).



Gambar 10.2: Struktur Numpy
Sumber: Diolah penulis

3. Demonstrasi Eksekusi & Pengujian Benchmark

Untuk membuktikan efisiensi komputasi NumPy secara empiris, berikut adalah pengujian komparatif penjumlahan 1.000.000 elemen menggunakan *looping* Python murni versus operasi tervektorisasi NumPy (*ufunc*) (Rougier, 2021; VanderPlas, 2022).

Daftar Pustaka

Gorelick, M., & Ozsvald, I. (2020). *High performance Python* (2nd ed.). O'Reilly Media, Inc.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., et al. (2020). Array programming with NumPy. *Nature, 585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering, 9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

McKinney, W. (2022). *Python for data analysis: Data wrangling with pandas, NumPy, and Jupyter* (3rd ed.). O'Reilly Media, Inc.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: *Machine learning* in Python. *Journal of Machine learning Research, 12*(85), 2825–2830. https://jmlr.org/papers/v12/pedregosa11a.html

Rougier, N. P. (2021). *From Python to NumPy*. Inria & Université de Bordeaux. https://doi.org/10.5281/zenodo.225783

Unpingco, J. (2014). *Python for signal processing: Featuring IPython notebooks*. Springer International Publishing. https://doi.org/10.1007/978-3-319-01342-8

VanderPlas, J. (2022). *Python data science handbook: Essential tools for working with data* (2nd ed.). O'Reilly Media, Inc.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., et al. (2020). SciPy 1.0: Fundamental algorithms for scientific *computing* in Python. *Nature Methods, 17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2

PROFIL PENULIS

Ir. Indo Intan, S.T., M.T.

Penulis menyelesaikan pendidikan sarjana tahun 2002 dan pendidikan magister tahun 2020 di Universitas Hasanuddin pada Program Studi Teknik Elektro. Terdaftar sebagai dosen di Universitas Dipa Makassar sejak tahun 2005 pada Prodi Teknik Informatika, Bidang *Coverage Artificial Intelligence,* di antaranya Pembelajaran Mesin, *Deep Learning*, dan Pengenalan Pola. Selain mengajar, penulis aktif melakukan kegiatan penelitian dan pengabdian masyarakat serta publikasi ilmiah.

Buku yang penulis pernah terbitkan di antaranya: Statistik, Teman Belajar dan Analisis Mahasiswa(2019); Foundations of *Machine learning* (2023); *Machine learning*: Konsep, Algoritma, dan Implementasi (2024); Deep Learning: Teori, Algoritma, dan Aplikasi (2025); *Machine learning*: Teori, Model, dan Implementasi - Sebuah Pendekatan Komprehensif (2025); Pemodelan dan Visualisasi Data (2025); Metodologi Penulisan Karya Ilmiah (2026); Deep Learning: Teori, Arsitektur, dan Aplikasi Modern (2026); dan buku Pengolahan Citra Digital: Teori, Algoritma, dan Implementasi Praktis (2026); yang berfokus pada lstatistik, logistic regression, backpropagation neural network, convolutional neural network, pengolahan citra.

Penulis menuangkan kepakarannya dalam riset maupun dalam karya buku agar bisa memberikan knowledge, skill, maupun insight kepada mahasiswa dan masyarakat secara umum tentang betapa pentingnya literasi di era sekarang di saat semua telah tergantikan oleh *Artificial Intelligence* (AI) dan terdistraksi oleh *screen time*. Semoga buku ini bisa membuka wawasan kita bersama, bahwa eksplorasi ilmu melalui buku akan melahirkan “energi baru” untuk menjadi bekal mengajar, meriset, dan memberikan kontribusi di tengah masyarakat sebagai jariyah pahala.

Email Penulis: indo.intan@undipa.ac.id/
https://www.youtube.com/@indointanchannel

BAB 11
STRUKTUR DATA PANDAS

Muhamad Yusup, S.Kom., M.Kom.
Universitas Raharja

Pengantar Series dan DataFrame

Di dalam ekosistem pemrograman Python untuk analisis data, Pandas merupakan pustaka (*library*) utama yang paling diandalkan oleh para pengembang, ilmuwan data, dan peneliti. Keunggulan utama Pandas terletak pada kemampuannya dalam mengelola dan memanipulasi data terstruktur secara cepat, intuitif, dan efisien. Fondasi dari seluruh operasi di dalam Pandas bertumpu pada dua struktur data fundamental, yaitu Series dan DataFrame (McKinney, 2021).

Pemahaman yang komprehensif mengenai konsep dasar kedua struktur data ini menjadi prasyarat mutlak sebelum seseorang melangkah ke tahap pengolahan dataset berskala besar, pembersihan data (*data cleaning*), maupun penerapan model kecerdasan buatan.

1. **Konsep Dasar Pandas Series (Array Satu Dimensi Berlabel)**

   Series dapat didefinisikan sebagai struktur data berdimensi satu (*one-dimensional array*) yang mampu menyimpan berbagai tipe data, seperti integer, *float*, *string*, objek Python, dan tipe data lainnya. Karakteristik paling pembeda antara Series di Pandas dengan list bawaan Python adalah adanya label indeks (*index labels*).

   a. Struktur Indeks dan Nilai

      Setiap elemen di dalam sebuah Series memiliki dua komponen utama, yaitu nilai aktual (*value*s) dan label penjelas (index). Secara default, Pandas akan memberikan indeks numerik berurutan mulai dari angka 0 hingga N-1 (di mana N adalah jumlah elemen). Namun, indeks ini dapat dikustomisasi menggunakan label *string* atau tanggal sesuai kebutuhan analisis.

   b. Homogenitas Data

      Meskipun dapat menampung berbagai tipe data secara umum, elemen-elemen di dalam satu objek Series biasanya dioptimalkan untuk memiliki tipe data yang seragam guna mempercepat proses komputasi berbasis NumPy.

   Ilustrasi Konseptual Series:

   Bayangkan sebuah kolom tunggal pada tabel Excel yang berisi data jumlah mahasiswa per fakultas

Indeks: `['Fakultas Teknik', 'Fakultas Ekonomi', 'Fakultas Ilmu Komputer']`
Nilai: `[1200, 2400, 3100]`

2. Konsep Dasar Pandas DataFrame (Tabel Dua Dimensi Berorientasi Kolom)

Jika Series dianalogikan sebagai sebuah vektor satu dimensi atau satu kolom tabel, maka DataFrame adalah representasi dari tabel data dua dimensi (*two-dimensional* tabular data) yang memiliki sumbu baris (axis=0) dan sumbu kolom (axis=1)

a. Struktur Relasional Tabular
   DataFrame menyerupai lembar kerja (spreadsheet) seperti Microsoft Excel, tabel basis data relasional SQL, atau matriks multivariat. Setiap kolom di dalam DataFrame pada dasarnya adalah sebuah objek Series, di mana seluruh kolom tersebut berbagi indeks baris yang sama.
b. Heterogenitas Kolom
   Berbeda dengan matriks NumPy tradisional yang mewajibkan seluruh elemen memiliki tipe data yang seragam, DataFrame bersifat heterogen antarkolom. Artinya, Kolom A dapat berisi tipe data integer, Kolom B berisi *string*, dan Kolom C berisi tipe data boolean atau datetime.
c. Fleksibiltas Indeks
   DataFrame dilengkapi dengan indeks baris yang fleksibel (bisa berupa angka, *string*, atau MultiIndex) serta nama kolom yang deskriptif, sehingga memudahkan proses pelacakan, pengelompokan (*grouping*), dan pemfilteran data.
d. Ilustrasi Konseptual DataFrame
   Sebuah tabel rekapitulasi data akademik yang terdiri dari beberapa kolom (Atribut):
   1) Kolom 1 (`NIM`): Tipe data *String* / Integer
   2) Kolom 2 (`Nama Mahasiswa`): Tipe data *String*
   3) Kolom 3 (`IPK`): Tipe data *Float*

Daftar Pustaka

Firdose, T. (2024). Ultimate Pandas for Data Manipulation and Visualization: Efficiently Process and Visualize Data with Python's Most Popular Data Manipulation Library. Orange Education Pvt Ltd.

McKinney, W. (2021). Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython (3rd ed.). O'Reilly Media.

Rerung, R. R., Fauzan, M., & Hermawan, H. (2020). Website Quality Measurement of Higher Education Services Institution Region IV Using Webqual 4.0 Method. International Journal of Advances in Data and Information Systems, 1(2), 89-102.

Sada, S. K. (2026). Arsitektur Algoritma dan Pengolahan Data Terstruktur Menggunakan Python Pandas. Penerbit Universitas Raharja.

Stewart, D., & Simmons, M. (2010). The Business Playground: Where Creativity and Commerce Collide. Berkeley, AS: New Riders Press.

Yildiz, M. (2024). Mastering Pandas: A Comprehensive Guide to Data Analysis in Python. Independently Published

PROFIL PENULIS

Muhamad Yusup, S.Kom.,M.Kom.

Ketertarikan penulis terhadap bidang ilmu komputer dimulai sejak tahun 2007. Minat tersebut mendorong penulis untuk menempuh pendidikan di SMAN 4 Tangerang dengan memilih jurusan Ilmu Pengetahuan Alam (IPA) dan berhasil menyelesaikan pendidikan menengah pada tahun yang sama. Selanjutnya, penulis melanjutkan studi ke jenjang pendidikan tinggi dan berhasil meraih gelar Sarjana Komputer (S.Kom.) dari Program Studi Sistem Informasi di STMIK Raharja. Tidak berhenti sampai di situ, penulis kemudian melanjutkan pendidikan magister di Program Studi Ilmu Komputer Universitas Budi Luhur dan berhasil meraih gelar Magister Komputer (M.Kom.) pada tahun 2014. Saat ini, penulis sedang menempuh pendidikan program doktor (S3) di Program Studi Ilmu Komputer Universitas Kristen Satya Wacana, Salatiga.

Penulis memiliki kepakaran di bidang Sains Data melalui workshop digitalent Departemen Kominfo RI selama 3 bulan. Saat ini penulis aktif sebagai dosen pada Program Studi Bisnis Digital Fakultas Ekonomi dan Bisnis Universitas Raharja. Dalam perannya sebagai akademisi, penulis terus mengembangkan keilmuan melalui kegiatan penelitian, pengajaran, dan penulisan ilmiah yang berfokus pada bidang kepakarannya. Penulis memiliki komitmen untuk berkontribusi dalam pengembangan teknologi dan pendidikan di Indonesia, khususnya dalam era transformasi digital.

Email Penulis: yusup@raharja.info

BAB 12
MANIPULASI DAN TRANSFORMASI DATA

Dewi Immaniar Desrianti, S.Kom., M.TI.
Universitas Raharja

Bab ini membahas konsep, teknik, dan implementasi manipulasi serta transformasi data menggunakan bahasa pemrograman Python. Materi difokuskan pada pengolahan berbagai struktur data, pembersihan data, pengubahan format, penggabungan, penyaringan, hingga transformasi data sebagai fondasi dalam analisis data dan pengembangan *machine learning*. Manipulasi dan transformasi data dengan Python adalah proses mengolah, mengubah, membersihkan, dan menyusun data menggunakan bahasa pemrograman Python agar data menjadi lebih mudah dianalisis dan digunakan

Konsep Dasar Manipulasi dan Transformasi Data

1. Pengertian Manipulasi Data

Manipulasi data adalah proses melakukan perubahan, pengaturan, pemilihan, penghapusan, atau pengolahan terhadap data untuk memenuhi kebutuhan tertentu. Manipulasi data dapat dilakukan pada berbagai jenis data, seperti angka, teks, daftar data, data mahasiswa, data transaksi, data sensor, dan dataset penelitian.

Contoh manipulasi data:

```python
nilai = [70, 80, 90]

nilai.append(100)
print(nilai)
```

Hasil:

```
[70, 80, 90, 100]
```

Pada contoh tersebut, data dimanipulasi dengan menambahkan nilai baru ke dalam daftar.

2. Pengertian Transformasi Data

Transformasi data adalah proses mengubah bentuk, format, tipe, struktur, atau nilai suatu data menjadi bentuk lain agar lebih sesuai untuk tujuan pengolahan dan analisis.

Contoh:

```python
nilai = "85"
nilai_angka = int(nilai)

print(nilai_angka)
```

Data yang awalnya berupa teks (*string*) ditransformasikan menjadi tipe data bilangan bulat (*integer*).

Transformasi data sering dilakukan untuk:

a. Menyesuaikan format data.
b. Menstandarkan nilai.
c. Mengubah data kategorikal menjadi angka.
d. Menyiapkan data untuk analisis.
e. Menyiapkan data untuk *machine learning*.

Tabel 12.1: Perbedaan Manipulasi dan Transformasi Data

Manipulasi Data	Transformasi Data
Mengelola atau mengubah data	Mengubah bentuk atau representasi data
Menambah data	Mengubah tipe data
Menghapus data	Melakukan normalisasi
Mengurutkan data	Mengubah kategori menjadi angka
Melakukan *filtering*	Melakukan standardisasi
Menggabungkan data	Membuat variabel baru

Sumber: Diolah penulis

Secara sederhana:

Manipulasi data berfokus pada pengelolaan data, sedangkan transformasi data berfokus pada perubahan bentuk atau representasi data.

3. Peran Pengolahan Data dalam Pemrograman Modern

Pengolahan data memiliki peranan penting dalam berbagai bidang, antara lain:

Daftar Pustaka

Géron, A. (2022). Hands-On *Machine learning* with Scikit-Learn, Keras, and TensorFlow (3rd ed.). O'Reilly Media.

Han, J., Pei, J., & Tong, H. (2023). Data Mining: Concepts and Techniques (4th ed.). Morgan Kaufmann.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). An Introduction to Statistical Learning: With Applications in R (2nd ed.). Springer.

McKinney, W. (2022). Python for Data Analysis: Data Wrangling with Pandas, NumPy, and Jupyter (3rd ed.). O'Reilly Media.

PROFIL PENULIS

Dewi Immaniar Desrianti, S.Kom., M.TI.

Merupakan dosen tetap pada Program Studi Teknik Informatika, Fakultas Sains dan Teknologi, Universitas Raharja. Beliau menyelesaikan pendidikan Sarjana (S.Kom.) bidang Teknik Informatika di STMIK Raharja pada tahun 2010 dan Magister (M.T.I.) bidang Teknik Informatika pada tahun 2015. Saat ini, beliau masih aktif sebagai dosen di Universitas Raharja. Selain menjalankan tugas akademik, beliau aktif dalam kegiatan pendidikan, penelitian, publikasi ilmiah, pengabdian kepada masyarakat, serta sebagai sekretaris redaksi jurnal ilmiah pada CCIT Journal. Selain itu, beliau juga aktif menjadi reviewer jurnal dan tim penguji/asesor kompetensi di bidang Teknologi Informatika dan Komunikasi. Bidang keahlian dan pengajaran yang ditekuni meliputi Pemrograman, Rekayasa Perangkat Lunak, *Machine Learning*, Data Warehouse, Mobile *Computing*, Komunikasi Data, Sistem Digital, Organisasi Komputer, Interaksi Manusia dan Komputer, serta *Computer Generated Imagery* (CGI). Email Penulis: dewi.immaniar@raharja.info

BAB 13
PEMBERSIHAN DAN PRA-PEMROSESAN DATA

Riri Fajriah, S.Kom., MM., M.Kom.
Universitas Mercu Buana

Pendahuluan

Data merupakan fondasi utama dalam pengembangan sistem komputasi modern, khususnya pada analisis data, kecerdasan buatan, dan *machine learning*. Algoritma pembelajaran mesin pada dasarnya mempelajari pola dari data yang tersedia. Oleh karena itu, kualitas data yang digunakan memiliki pengaruh langsung terhadap kualitas pola yang dapat dipelajari oleh model. Data yang mengandung nilai hilang, duplikasi, kesalahan pencatatan, ketidakkonsistenan format dan fitur yang tidak relevan dapat menyebabkan proses pembelajaran menghasilkan model yang kurang akurat dan sulit digeneralisasikan. Menurut Geron (2022), kualitas data merupakan salah satu tantangan utama dalam proyek *machine learning*. Sementara itu, menurut Huyen (2022), ditekankan bahwa sistem *machine learning* sangat bergantung pada karakteristik data yang digunakan.

Dalam praktiknya, data mentah (*raw data*) hampir tidak pernah langsung siap digunakan untuk algoritma *machine learning*. Data dapat diperoleh dari berbagai sumber, seperti basis data relasional, file CSV, *spreadsheet*, API, sensor, aplikasi web, dan media sosial, yang merupakan sistem informasi organisasi. Perbedaan sumber tersebut dapat menyebabkan perbedaan struktur, tipe data, satuan pengukuran, format tanggal, aturan penamaan atribut, hingga tingkat kelengkapan data. Dengan demikian, sebelum data digunakan untuk membangun model, diperlukan serangkaian aktivitas sistematis yang dikenal sebagai *data cleaning* dan *data preprocessing* atau pembersihan dan pra-pemrosesan data.

McKinney (2022) menjelaskan bahwa pengolahan data dengan Python melibatkan berbagai aktivitas seperti memuat, membersihkan, mentransformasi, menggabungkan, membentuk ulang, dan menganalisis dataset. Python menyediakan ekosistem yang sangat kuat untuk aktivitas tersebut melalui pustaka seperti NumPy, pandas, Matplotlib, dan scikit-learn. Penggunaan pustaka tersebut memungkinkan proses pembersihan dan prapemrosesan dilakukan secara terstruktur dan dapat direproduksi.

Pembersihan data berfokus pada identifikasi dan perbaikan masalah kualitas data, sedangkan pra-pemrosesan memiliki cakupan

yang lebih luas, yaitu mengubah data yang telah dibersihkan menjadi representasi yang sesuai dengan kebutuhan algoritma. Tahap ini dapat mencakup imputasi nilai hilang, penanganan pencilan, transformasi data, pengkodean variabel kategorikal, penskalan fitur, seleksi fitur, rekayasa fitur, reduksi dimensi, serta pembagian dataset menjadi data pelatihan dan pengujian. Raschka et al. (2022) menempatkan *preprocessing* sebagai bagian penting dalam *machine learning workflow* karena data harus terlebih dahulu berada dalam bentuk yang sesuai sebelum digunakan untuk pelatihan model.

Bab ini membahas konsep, prinsip, teknik, dan implementasi pembersihan serta prapemrosesan data menggunakan Python. Pembahasan diarahkan agar pembaca tidak hanya memahami sintaks Python, tetapi juga mampu mengambil keputusan analitis mengenai kapan suatu teknik digunakan, mengapa teknik tersebut diperlukan, dan bagaimana menghindari kesalahan dalam proses prapemrosesan.

Konsep Dasar Pembersihan Data

Pembersihan data (*data cleaning*) merupakan proses mendeteksi, mengevaluasi, memperbaiki, atau menghapus data yang tidak sesuai dengan kebutuhan analisis. Data bermasalah tidak selalu berarti data tersebut salah secara mutlak. Suatu nilai dapat dianggap bermasalah apabila tidak sesuai dengan konteks analisis, aturan bisnis, domain pengetahuan, atau kebutuhan algoritma.

Sebagai contoh, nilai usia sebesar 250 tahun hampir pasti merupakan kesalahan pencatatan untuk dataset manusia. Namun, nilai pendapatan sebesar Rp. 250.000.000 tidak dapat langsung dianggap sebagai pencilan atau kesalahan karena mungkin merupakan nilai yang benar. Oleh sebab itu, pembersihan data bukan sekadar proses menghapus nilai yang terlihat ekstrem, tetapi membutuhkan pemahaman terhadap konteks data.

Secara umum, masalah kualitas data dapat dikelompokkan menjadi :

1. Missing *value*s atau nilai hilang;
2. Data duplikat;
3. Kesalahan tipe data;

Daftar Pustaka

Gagolewski, M. (2023). *Minimalist data wrangling with Python.* Zenodo. https://doi.org/10.5281/zenodo.6451068

Geron, A. (2022). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems* (3rd ed.). O'Reilly Media.

Huyen, C. (2022). *Designing machine learning systems: An iterative process for production-ready applications.* O'Reilly Media.

James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). *An introduction to statistical learning: With applications in Python.* Springer. https://doi.org/10.1007/978-3-031-38747-0

McKinney, W. (2022). *Python for data analysis: Data wrangling with pandas, NumPy, and Jupyter* (3rd ed.). O'Reilly Media.

Rafatirad, S., Homayoun, H., Chen, Z., & Pudukotai Dinakarrao, S.M. (2022). *Machine learning for computer scientists and data analysts: From an applied perspective. Springer.*

Raschka, S., Liu, Y. (Hayden), Mirjalili, V., & Dzhulgakov, D. (2022). *Machine learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python. Packt Publishing.*

PROFIL PENULIS

Riri Fajriah, S.Kom, MM, M.Kom.

Penulis adalah seorang profesional dan akademisi yang memiliki keahlian di bidang tata kelola teknologi informasi dan data science. Beliau meraih gelar Sarjana Komputer pada tahun 2007 dari Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Bina Nusantara. Setelah menyelesaikan studi sarjananya, Riri memulai karier sebagai IT profesional di PT BNI Life Insurance, di mana posisi terakhir yang diembannya pada tahun 2016 adalah *Senior Assistant Manager Procurement*. Selama di perusahaan tersebut, beliau banyak terlibat dalam proyek pengadaan dan pengelolaan infrastruktur teknologi informasi. Pada tahun 2010, beliau melanjutkan pendidikan dan meraih gelar Magister Manajemen dari Universitas Mercu Buana. Untuk memperdalam keahliannya di bidang teknologi, Riri menyelesaikan program Magister Ilmu Komputer di Universitas Budi Luhur pada tahun 2024. Saat ini, beliau menjabat sebagai Dosen Tetap pada Program Studi Sistem Informasi Fakultas Ilmu Komputer Universitas Mercu Buana. Dalam perannya sebagai akademisi, Riri aktif melakukan penelitian yang berfokus pada tata kelola teknologi informasi dan data science. Penelitiannya telah memberikan kontribusi signifikan terhadap pengembangan ilmu di bidang tersebut, serta mendukung inovasi dalam pengelolaan TI dan analisis data di lingkungan akademik maupun industri.

Email Penulis : riri.fajriah@mercubuana.ac.id

BAB 14
VISUALISASI DATA ILMIAH

Oleh Soleh, S.Kom., M.MSI
Universitas Raharja

Di sebuah ruang kerja yang sunyi, seorang analis data menatap layar berisi jutaan baris angka, hasil transaksi, sensor, atau catatan eksperimen yang tampak tak berujung. Mata manusia tidak diciptakan untuk membaca lautan angka semacam itu; namun otak yang sama akan langsung "menangkap" sebuah pola begitu angka-angka itu berubah menjadi kurva, batang, atau titik-titik yang tersebar dalam sebuah grafik. Di titik inilah keajaiban visualisasi data terjadi: ia menerjemahkan bahasa dingin komputasi menjadi bahasa yang paling purba dimengerti manusia, bahasa gambar.

Bab ini mengajak pembaca melangkah lebih jauh dari sekadar menulis kode, menuju kemampuan untuk *melihat* apa yang tersembunyi di balik data, menemukan pola musiman yang tak terduga, mengenali outlier yang mencurigakan, atau menangkap kilasan wawasan yang kelak menjadi pijakan bagi model *machine learning* yang akan dibangun pada bab-bab berikutnya. Sebab pada akhirnya, data yang tidak pernah "dilihat" adalah data yang kehilangan separuh maknanya.

Pendahuluan

Dalam dunia yang dibanjiri oleh data (baik dari sensor, transaksi digital, media sosial, maupun eksperimen ilmiah), kemampuan untuk *melihat* pola di balik angka menjadi keterampilan yang tidak kalah penting dibandingkan dengan kemampuan menghitungnya. Visualisasi data bukan sekadar tahap "mempercantik" laporan di akhir sebuah analisis, melainkan komponen inti yang menyatu di sepanjang siklus kerja data *science*, mulai dari eksplorasi data mentah hingga penyampaian hasil kepada pengambil keputusan. Panghal (2024) menegaskan bahwa visualisasi data krusial bagi pengambilan keputusan yang tepat di dunia yang berorientasi pada data saat ini, karena mengubah kumpulan data yang kompleks menjadi representasi visual yang menarik sekaligus meningkatkan proses kognitif dan menumbuhkan wawasan.

Secara garis besar, siklus tersebut dapat digambarkan sebagai rangkaian: *Exploratory Data Analysis* (EDA) → penemuan wawasan (*insight*) → komunikasi hasil → pengambilan keputusan. Pada tahap

EDA, visualisasi berfungsi sebagai alat bantu analis untuk "mengenal" datanya, mengidentifikasi distribusi, outlier, dan hubungan antarvariabel sebelum masuk ke tahap pemodelan yang lebih formal. Shinde dan Shivthare (2024) secara khusus menyoroti keterkaitan visualisasi dengan performa model *machine learning*, dengan menjelaskan bahwa teknik visual meningkatkan berbagai aspek tahap analisis data (termasuk EDA, seleksi dan rekayasa fitur, deteksi anomali, serta validasi asumsi) yang pada akhirnya mengarah pada pengembangan model *machine learning* yang lebih akurat dan efisien. Temuan ini memperkuat posisi bab ini sebagai fondasi sebelum pembaca mempelajari algoritma ML pada Bab 18

Setelah wawasan ditemukan, peran visualisasi bergeser dari alat eksplorasi pribadi analis menjadi medium komunikasi kepada audiens yang lebih luas, termasuk dalam konteks pengambilan keputusan berbasis kecerdasan buatan yang semakin umum saat ini. Neri et al. (2025), melalui tinjauan sistematis terhadap 127 studi yang diterbitkan antara tahun 2011 hingga 2024, mengkaji pemanfaatan, tantangan, dan prinsip desain pendekatan visualisasi data dengan berfokus pada penerapannya dalam konteks pengambilan keputusan berbasis AI, termasuk mengidentifikasi elemen visual kunci yang memengaruhi pemahaman pengguna. Hasil tinjauan tersebut menunjukkan bahwa efektivitas visualisasi tidak berdiri sendiri, melainkan sangat bergantung pada bagaimana elemen visual dirancang agar selaras dengan cara kerja kognisi manusia, sebuah prinsip yang juga ditegaskan oleh riset di ranah kesehatan. Fusco et al. (2025) menjelaskan bahwa ketika transisi pemrosesan visual berlangsung mulus, upaya kognitif menjadi minimal dan pemahaman menjadi efisien, sementara visual yang dirancang buruk dapat memicu kesan keliru akibat penggunaan warna atau bentuk yang tidak konsisten. Studi ini relevan lintas domain: baik dalam visualisasi data bisnis, ilmiah, maupun klinis, prinsip kognitif yang sama tetap berlaku.

Namun demikian, riset mutakhir juga mencatat bahwa efektivitas visualisasi bukan jaminan otomatis. Dimara et al. (2022) menemukan bahwa kebutuhan visualisasi para pengambil keputusan di dalam organisasi sering kali belum terpenuhi secara optimal oleh perangkat visualisasi yang tersedia saat ini, karena kompleksitas konteks

Daftar Pustaka

Baniecki, H., & Biecek, P. (2019). modelStudio: Interactive studio with explanations for ML predictive models. *arXiv preprint*. https://arxiv.org/abs/2005.00497

Dimara, E., Zhang, H., Tory, M., & Franconeri, S. (2022). The unmet data visualization needs of decision makers within organizations. *IEEE Transactions on Visualization and Computer Graphics, 28*(12), 4101–4112. https://doi.org/10.1109/TVCG.2021.3074023

Fusco, R., Granata, V., Setola, S. V., Pupo, D., Petrosino, T., Lamanna, C. P., Castaldo, M., Riga, M. G., Karaboue, M. A., Izzo, F., & Petrillo, A. (2025). Visual perception and pre-attentive attributes in oncological data visualisation. *Bioengineering, 12*(7), 782. https://doi.org/10.3390/bioengineering12070782

National Academy of Sciences. (2023). *Digital art guidelines*. PNAS. https://www.pnas.org/pb-assets/authors/digitalart-1675347574760.pdf

Neri, G., Marshall, S., Chan, H. K.-H., Yaghi, A., Tabor, D., Sinha, R., & Mazumdar, S. (2025). Data visualization in AI-assisted decision-making: A systematic review. *Frontiers in Communication, 10*, 1605655. https://doi.org/10.3389/fcomm.2025.1605655

Nuñez, J. R., Anderton, C. R., & Renslow, R. S. (2018). Optimizing colormaps with consideration for color vision deficiency to enable accurate interpretation of scientific data. *PLOS ONE, 13*(7), e0199239. https://doi.org/10.1371/journal.pone.0199239

Panghal, R. (2024). The role of data visualization in decision making: Case of D-mart. *International Journal for Multidisciplinary Research, 6*(3). https://doi.org/10.36948/ijfmr.2024.v06i03.19630

Roberts, J., Butcher, P., Sherlock, A., & Nason, S. (2021). Explanatory journeys: Visualising to understand and explain administrative justice paths of redress. *arXiv preprint*. https://arxiv.org/abs/2107.14013

Rougier, N. P. (2021). *Scientific visualization: Python + Matplotlib* (1st ed.). AFNIL. https://doi.org/10.5281/zenodo.4638720

Shinde, B. G., & Shivthare, S. (2024). Impact of data visualization in data analysis to improve the efficiency of *machine learning* models. *Journal of Advanced Zoology, 45*(S4).

Statistics Canada. (2023). *Data visualization: Best practices*. Government of Canada. https://www150.statcan.gc.ca/n1/pub/89-26-0005/892600052022001-eng.htm

Waskom, M. L. (2021). Seaborn: Statistical data visualization. *Journal of Open Source Software, 6*(60), 3021. https://doi.org/10.21105/joss.03021

PROFIL PENULIS

Oleh Soleh S.Kom., M.MSI

Ketertarikan penulis terhadap ilmu komputer dimulai pada tahun 1996 silam. Hal tersebut membuat penulis memilih untuk masuk ke Universitas Gunadarma dengan memilih Jurusan Manajemen Informatika (MI) pada Fakultas Ilmu Komputer dan berhasil menyelesaikan program studi S1 dengan kelulusan pada tahun 2000. Penulis kemudian melanjutkan pendidikan ke jenjang master di kampus yang sama. Pada tahun 2004, penulis berhasil menyelesaikan studi S2 di Prodi Ilmu Komputer dengan Jurusan Manajemen Informasi Bisnis. Mulai tahun 2022 sampai saat ini sedang menempuh pendidikan doktoral di Universitas Kristen Satya Wacana Salatiga melalui prodi Ilmu Komputer dengan konsentrasi *Deep Learning & Computer Vision.* Penulis memiliki kepakaran di bidang *Business Intelligence*, *Data Mining,* Manajemen Informasi serta *Deep Learning*. Dan untuk mewujudkan karier sebagai dosen profesional, penulis pun aktif sebagai peneliti di bidang kepakarannya tersebut. Beberapa penelitian yang telah dilakukan didanai oleh pihak internal perguruan tinggi.

Email Penulis: oleh.soleh@raharja.info

BAB 15
ALGORITMA *MACHINE LEARNING* KLASIK

Falahah, S.Si., M.T.
Universitas Telkom

Machine learning

Machine learning (selanjutnya dapat disingkat sebagai ML) atau pembelajaran mesin dapat didefinisikan sebagai suatu cabang dari kecerdasan buatan yang memungkinkan komputer untuk belajar dari data dan pengalaman masa lampau, tanpa harus dinyatakan secara eksplisit dalam bentuk program. Bidang *machine learning* meliputi statistik, ilmu komputer, dan teori optimasi untuk mendeteksi pola-pola tertentu yang dianggap penting pada data (Alshammari et al., 2026).

Saat ini ML menjadi teknologi utama untuk tugas-tugas tertentu seperti klasifikasi, regresi, pengenalan pola, prediksi, penambangan data (*data mining*), pengolahan citra, dan otomatisasi keputusan pada data dengan dimensi tinggi. Ruang lingkup aplikasi ML misalnya mencakup diagnosis medis, pengenalan suara dan gambar, sistem rekomendasi berdasarkan perilaku pengguna, penyaringan spam, robotika, kendaraan otonom, hingga bioinformatika (Alshammari et al., 2026; Vaswani et al., 2017).

ML sendiri sudah mengalami sejarah panjang sejak awal dirintis pada tahun 1950 ketika Alan Turing mengenalkan konsep "*Computing Machinery and Intelligence*" yang menjelaskan ide bahwa mesin dapat berpikir, hingga kondisi terkini dengan munculnya *Generative AI* yang kemudian diikuti oleh Agentic AI. Gambar 18.1 menampilkan linimasa perkembangan ML dari awal hingga sekarang. (Alshammari et al., 2026; Vaswani et al., 2017)

Gambar 15.1: Linimasa Perkembangan *Machine Learning*
Sumber: Diolah penulis

Daftar Pustaka

Alshammari, A. H., Bencsik, G., & Ali, A. H. (2026). A survey of six *class*ical *class*ifiers, including algorithms, methodological characteristics, foundational variants, and recent advances. Algorithms, 19(1), 37. https://doi.org/10.3390/a19010037

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785–794). Association for *Computing* Machinery. https://doi.org/10.1145/2939672.2939785

Emmanuel, T., Maupong, T., Mpoeleng, D., Semong, T., Mphago, B., & Tabona, O. (2021). A survey on missing data in *machine learning*. Journal of Big Data, 8, 140. https://doi.org/10.1186/s40537-021-00516-9

Ezugwu, A. E., Ikotun, A. M., Oyelade, O. O., Abualigah, L., Agushaka, J. O., Eke, C. I., & Akinyelu, A. A. (2022). A comprehensive survey of clustering algorithms: State-of-the-art *machine learning* applications, taxonomy, challenges, and future research prospects. Engineering Applications of Artificial Intelligence, 110, 104743. https://doi.org/10.1016/j.engappai.2022.104743

Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: A review and recent developments. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 374(2065), 20150202. https://doi.org/10.1098/rsta.2015.0202

Kaur, H., Pannu, H. S., & Malhi, A. K. (2019). A systematic review on imbalanced data challenges in *machine learning*: Applications and solutions. ACM *Computing* Surveys, 52(4), Article 79. https://doi.org/10.1145/3343440

Pintas, J. T., Fernandes, L. A. F., & Garcia, A. C. B. (2021). Feature selection methods for text *class*ification: A systematic literature review. Artificial Intelligence Review, 54, 6149–6200. https://doi.org/10.1007/s10462-021-09970-6

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In

Advances in Neural Information Processing Systems (Vol. 30). https://doi.org/10.48550/arXiv.1706.03762

PROFIL PENULIS

Falahah, S.Si., M.T.

Penulis adalah seorang dosen dan praktisi di bidang ilmu komputer, yang sudah berkecimpung di bidang industri dan pendidikan sejak 30 tahun yang silam. Setelah menyelesaikan pendidikan S1 di bidang sains, penulis kemudian bekerja sebagai analis sistem pada divisi IT sebuah BUMN selama tujuh tahun. Kemudian, penulis melanjutkan studi S2 di bidang Sistem Informasi tahun 2006, kemudian aktif terjun sebagai pengajar di berbagai perguruan tinggi di Bandung dan sekitarnya. Selain sebagai akademisi, penulis juga aktif terlibat dalam kegiatan jasa konsultasi di bidang IT di berbagai perusahaan maupun lembaga pemerintahan. Penulis juga aktif menghasilkan karya akademis berupa artikel pada beberapa jurnal dan seminar nasional/internasional, serta menulis beberapa buku di bidang basis data, rekayasa perangkat lunak, serta *Enterprise Resource Planning* (ERP). Selain pada bidang *machine learning* dan kecerdasan buatan, penulis juga memiliki minat yang tinggi pada bidang tata kelola teknologi informasi, arsitektur enterprise, manajemen layanan, manajemen proyek dan manajemen risiko di bidang teknologi informasi. Informasi dan komunikasi lebih lanjut dengan penulis dapat melalui alamat email: andromeda1268@gmail.com

ALGORITMA DAN PEMROGRAMAN PYTHON MODERN

Teori, Struktur Data, dan Fondasi Komputasi **Machine Learning**

Perkembangan teknologi komputasi dalam satu dekade terakhir telah mengubah lanskap ilmu pengetahuan secara fundamental. Python, sebagai salah satu bahasa pemrograman yang paling pesat berkembang, telah menjadi *lingua franca* bagi para ilmuwan data, pengembang perangkat lunak, dan peneliti di seluruh dunia. Kehadiran Python tidak hanya mempermudah proses pembuatan program, tetapi juga menjadi jembatan antara konsep algoritma klasik dengan kebutuhan komputasi modern, terutama di bidang *machine learning* dan kecerdasan artifisial. Buku ini disusun dengan kesadaran bahwa pembelajaran algoritma dan pemrograman tidak cukup hanya berhenti pada sintaksis bahasa. Lebih dari itu, pembaca perlu memahami bagaimana sebuah algoritma dirancang, bagaimana data disimpan dan dimanipulasi melalui struktur data yang tepat, serta bagaimana fondasi komputasi tersebut menjadi pijakan dalam membangun model *machine learning* yang efisien dan akurat. Oleh karena itu, buku ini dibagi ke dalam tiga pilar utama. Pertama, pembahasan teori algoritma dan pemrograman Python modern, yang mencakup dasar-dasar logika, kontrol alur, fungsi, hingga paradigma pemrograman berorientasi objek dan fungsional. Kedua, eksplorasi struktur data mulai dari senarai, tumpukan, antrean, pohon, graf, hingga struktur data lanjutan yang disajikan dengan implementasi langsung dalam Python. Ketiga, fondasi komputasi untuk machine learning, yang membahas dasar aljabar linear, statistika komputasi, vektorisasi, serta pengenalan *library* kunci seperti NumPy, pandas, dan scikit-learn.

sada kurnia pustaka

ISBN 978-634-7949-01-1