E-ISSN : 3025-1311
https://ejournal.warunayama.org/kohesi

Kohesi: Jurnal Multidisiplin Saintek
Volume 9 No 2 Tahun 2025

ANALISIS PERAN ENKAPSULASI DALAM BAHASA PEMROGRAMAN PYTHON TERHADAP KEAMANAN DAN REUSABILITY KODE

Wahyu Rahman Hakim[1], Josefin Anderson [2], Yuda Al'Hadid[3], M. Arya Saputra[4], Gunawan Saputra[5]

[1, 2,3,4,5] Program Studi Teknik Informatika, Fakultas Teknik, Universitas Muhammadiyah Bengkulu,
Jl. Bali, Kampung Bali, Teluk Segara, Kota Bengkulu, Bengkulu, Indonesia
Email Korespondensi: waraha93@gmail.com

A B S T R A K

Enkapsulasi merupakan konsep inti dalam Pemrograman Berorientasi Objek (PBO) yang berperan penting dalam meningkatkan keamanan serta kemampuan penggunaan ulang (reusabilitas) kode. Tidak seperti Java atau C++ yang menerapkan kontrol akses secara ketat, Python menggunakan konvensi penamaan (seperti awalan garis bawah) dan dekorator @property untuk mengatur visibilitas data. Penelitian ini mengkaji bagaimana mekanisme enkapsulasi di Python berkontribusi dalam menjaga keamanan data internal serta mendukung struktur kode yang modular dan mudah dipelihara. Dengan metode deskriptif kualitatif melalui studi pustaka, penelitian ini menganalisis praktik enkapsulasi berdasarkan literatur akademik, dokumentasi resmi, dan artikel teknis. Hasilnya menunjukkan bahwa meskipun Python tidak membatasi akses secara teknis, enkapsulasi tetap memberikan manfaat nyata dalam membatasi akses langsung terhadap atribut objek, menjaga konsistensi data, dan mempermudah penggunaan ulang kode. Tantangan yang ditemukan antara lain fleksibilitas Python yang membuat pemula cenderung mengabaikan prinsip enkapsulasi, serta celah akses melalui teknik name mangling. Penelitian ini memberikan saran praktis seperti penggunaan @property, perancangan antarmuka publik yang jelas, dan penguatan modularitas kode. Secara keseluruhan, penerapan enkapsulasi yang tepat dalam Python tetap penting untuk menghasilkan sistem perangkat lunak yang aman dan dapat digunakan kembali.

Kata kunci: enkapsulasi, keamanan perangkat lunak, modularitas, Python, reusabilitas kode

A B S T R A C T

Encapsulation is a core concept of Object-Oriented Programming (OOP) that plays a critical role in enhancing both security and code reusability. Unlike languages such as Java or C++ that apply strict access control, Python utilizes naming conventions (e.g., underscore prefixes) and the @property decorator to manage data visibility. This study investigates how encapsulation mechanisms in Python contribute to securing internal data and promoting modular, maintainable code. Employing a qualitative descriptive approach through literature review, the research

Article History
Received: Juni 2025
Reviewed: Juni 2025
Published: Juli 2025

Plagirism Checker No 234
Prefix DOI : Prefix DOI : 10.8734/Kohesi.v1i2.365
Copyright : Author
Publish by : Kohesi

draws from academic sources, documentation, and technical articles to analyze encapsulation practices. The findings indicate that, although Python does not enforce access restrictions technically, encapsulation still offers significant benefits by minimizing direct access to object attributes, ensuring data consistency, and simplifying code reuse. Challenges identified include Python's flexible structure that may lead beginners to overlook encapsulation principles, and the ease of bypassing data protection via name mangling. To address this, the study provides practical recommendations such as applying @property, defining clear public interfaces, and reinforcing code modularity. Overall, encapsulation in Python, when applied properly, remains essential for developing secure and reusable software systems.

***Keywords**: code reusability, encapsulation, modularity, Python, software security*

PENDAHULUAN

Paradigma Pemrograman Berorientasi Objek (PBO) menjadi pendekatan dominan dalam rekayasa perangkat lunak di era pengembangan perangkat lunak yang semakin kompleks karena memungkinkan pembuatan kode yang modular, dapat digunakan kembali, dan lebih mudah dirawat (Martin, 2008). Dalam survei besar mengenai paket-paket Python di repositori publik PyPI, ditemukan bahwa sekitar 46% paket memiliki setidaknya satu masalah keamanan saat dianalisis secara statis, kode atau modul Python rentan terhadap ancaman, termasuk penanganan exception, injeksi, dan celah modul seperti subprocess(Ruohonen et al., 2021). Salah satu elemen krusial dalam PBO adalah enkapsulasi, yaitu metode yang membatasi akses langsung ke atribut atau metode dalam objek, sehingga hanya bisa diakses melalui interface tertentu. Enkapsulasi bertujuan untuk melindungi data internal, mencegah manipulasi yang tidak diinginkan, serta meningkatkan konsistensi dan keandalan sistem (Ali et al., 2024).

Dalam bahasa pemrograman seperti Java dan C++, mekanisme enkapsulasi diterapkan melalui sistem pembatasan akses yang ketat (seperti private, protected, dan public). Akan tetapi, Python mengadopsi cara yang lebih santai, karena tidak menawarkan sistem kontrol akses yang ketat secara teknis, melainkan mengandalkan konvensi penamaan atribut (seperti prefiks underscore _ atau __) dan fitur @property untuk mengelola visibilitas serta pengendalian data Click or tap here to enter text.. Dari kajian pustaka, banyak penelitian menyoroti keamanan Python dengan teknik-statis atau enkripsi, tetapi tidak secara khusus menelaah peran enkapsulasi sebagai blok pembangun struktur kode yang aman dan reusable. Kesenjangan penelitian inilah yang mendorong perlunya fokus khusus pada enkapsulasi, pendekatan ini memunculkan pertanyaan: apakah enkapsulasi di Python cukup efektif untuk melindungi keamanan data internal dan mendukung penggunaan kembali kode dalam konteks rekayasa perangkat lunak yang professional.

Dari permasalahan itu, penelitian ini bertujuan untuk menganalisis penggunaan enkapsulasi dalam bahasa Python serta mengevaluasi kontribusinya terhadap aspek keamanan dan penggunaan kembali kode program. Selain itu, studi ini juga bertujuan untuk menghasilkan rekomendasi desain yang aplikatif dan relevan berdasarkan prinsip enkapsulasi, untuk meningkatkan kualitas kode Python dalam hal keamanan dan kemudahan pemeliharaan (maintainability). Inti dari penelitian ini adalah cara konvensi dan mekanisme enkapsulasi

diterapkan dalam Python, serta bagaimana mekanisme tersebut membantu dalam membangun struktur kode yang aman dan mudah untuk digunakan kembali

Studi ini diharapkan dapat memberikan sumbangan secara teoritis, yaitu meningkatkan pemahaman dan referensi mengenai praktik enkapsulasi dalam paradigma PBO melalui pendekatan Python, serta kontribusi praktis, yaitu menjadi acuan untuk mahasiswa dan pengembang perangkat lunak pemula dalam menyusun kode Python yang lebih terstruktur, aman, dan modular

METODE

Susunan umum makalah

Penelitian ini menerapkan metode deskriptif melalui pendekatan studi pustaka (library research). Metode deskriptif bertujuan untuk menyajikan secara sistematis dan faktual tentang ciri-ciri fenomena yang sedang diteliti (Avianti et al., 2010). Pendekatan studi pustaka merupakan metode penelitian yang memanfaatkan sumber-sumber tertulis seperti buku, jurnal, artikel, dan dokumen lainnya untuk mengumpulkan data serta informasi yang berhubungan dengan topik penelitian (Del Cid et al., 2009). dalam hal ini adalah fungsi enkapsulasi dalam bahasa pemrograman Python terhadap aspek keamanan dan reusabilitas kode.

Jenis penelitian ini merupakan penelitian deskriptif kualitatif yang menekankan pada pemahaman yang mendalam mengenai konsep dan praktik enkapsulasi dalam Python (Adlini et al., 2022). Metode yang diterapkan adalah studi literatur, yaitu cara pengumpulan data dan informasi yang berasal dari berbagai referensi tertulis.

Sumber data dalam penelitian ini bersifat sekunder, yang terdiri dari:

1. Buku teks tentang Pemrograman Berorientasi Objek (PBO)
2. Dokumentasi resmi Python (https://docs.python.org)
3. Jurnal ilmiah nasional dan internasional
4. Artikel teknis dari situs terverifikasi seperti RealPython, GeeksForGeeks, dan DataCamp

Proses pengumpulan data dilakukan dengan mencari referensi-referensi yang terkait dengan topik enkapsulasi pada Python. Beberapa kata kunci yang dipakai dalam proses pencarian meliputi: "Encapsulation in Python", "Python OOP", "data hiding in Python", "keamanan dalam kode Python", dan "reuse kode di Python"

Analisis data dilakukan secara tematik kualitatif, dengan langkah-langkah sebagai berikut:

1. Mengidentifikasi dan mengklasifikasikan data, yaitu mengelompokkan referensi menurut tema utama (konsep enkapsulasi, metode implementasi, keamanan, dan reusabilitas).
2. Analisis isi, yaitu menelaah, memahami, dan menginterpretasikan konten bahan pustaka untuk menemukan makna dan keterkaitan antara konsep-konsep utama.
3. Analisis data, yaitu menyusu n penjelasan atau argumen berdasarkan hasil yang didapat, lalu menghubungkannya dengan praktik rekayasa perangkat lunak yang sesuai.

Hasil dari proses ini disajikan secara naratif untuk menjelaskan bagaimana enkapsulasi dalam Python mendukung keamanan data internal serta meningkatkan efisiensi dalam pemanfaatan kembali kode pada pengembangan perangkat lunak.

Dalam menjaga keabsahan data, penelitian ini hanya memanfaatkan sumber-sumber yang terpercaya, seperti jurnal ilmiah yang terindeks, buku-buku akademik, dan dokumentasi resmi Python. Literatur yang dipilih juga disaring berdasarkan keterkaitan dengan topik penelitian serta keaktualan kontennya, agar cocok dengan konteks perkembangan teknologi perangkat lunak saat ini.

HASIL DAN DISKUSI

Enkapsulasi merupakan konsep fundamental dalam pemrograman berorientasi objek yang bertujuan menutupi rincian internal sebuah objek dan hanya memperlihatkan elemen yang

diperlukan lewat antarmuka publik. Dalam Python, konsep ini diterapkan bukan melalui pengaturan akses seperti private, protected, atau public seperti di Java atau C++, melainkan dengan menggunakan konvensi penamaan (naming convention) seperti:

1. Satu garis bawah (_) digunakan untuk menunjukkan atribut sebagai "protected" (dalam konteks informal)
2. Dua garis bawah (__) untuk "name mangling" yang menyulitkan akses eksternal secara langsung
3. Fitur @property dan @setter untuk mengatur akses baca-tulis atribut dengan cara yang terkontrol

Contoh implementasi enkapsulasi Python bisa dilihat pada gambar dibawah ini.

```python
class User:
    def __init__(self, username):
        self.__username = username

    @property
    def username(self):
        return self.__username

    @username.setter
    def username(self, value):
        if isinstance(value, str):
            self.__username = value
```

Gambar 1. Contoh Enkapsulasi pada Python

Gambar 1 memperlihatkan sebuah kelas Python yang diberi nama User dengan atribut privat __username. Atribut ini dirancang agar tidak dapat diakses langsung dari luar kelas karena menggunakan double underscore (__), yang menerapkan teknik name mangling dalam Python. Untuk memberikan akses ke atribut tersebut, digunakan dekorator @property, yang memungkinkan username diakses layaknya atribut biasa namun dengan cara yang teratur. Selanjutnya, ada pula metode @username.setter yang berfungsi untuk menetapkan atau mengubah nilai __username. Di dalam setter itu, ditambahkan validasi tipe data untuk memastikan bahwa nilai yang diberikan adalah string, sehingga data tetap terjaga konsistensinya dan aman. Mekanisme seperti ini menggambarkan prinsip enkapsulasi dalam OOP, di mana akses terhadap data internal dibatasi dan dikelola melalui metode publik yang aman.

Enkapsulasi memungkinkan pengembang untuk menyembunyikan data internal agar tidak dapat diakses secara ilegal oleh pihak luar. Dengan mekanisme __attribute dan pengaturan setter, programmer dapat:

1. Melindungi karakteristik dari modifikasi langsung
2. Melakukan validasi sebelum merubah nilai
3. Menghindari penyalahgunaan objek oleh modul lain

Dalam studi oleh Ruohonen et al. (2021), diketahui bahwa 46% paket Python di PyPI memiliki kemungkinan adanya celah keamanan. Banyak kerentanan ini muncul akibat kode yang tidak terproteksi dengan baik, seperti penggunaan atribut publik tanpa pembatasan akses [2]. Oleh sebab itu, penerapan enkapsulasi yang tepat sangat berkontribusi dalam mengurangi risiko tersebut.

Enkapsulasi juga memiliki peran penting dalam mendukung modularitas serta penggunaan ulang kode. Dengan membatasi akses hanya lewat metode publik (antarmuka), komponen kode menjadi lebih tertutup, modular, dan lebih mudah untuk diuji. Hal ini sangat bermanfaat dalam praktik pengembangan perangkat lunak yang melibatkan tim dan pemanfaatan pustaka kode. Misalnya, dalam pengembangan pustaka (library) Python, penerapan enkapsulasi dapat memastikan bahwa hanya fungsi-fungsi tertentu yang terlihat, sehingga menghindari ketergantungan eksternal pada komponen internal program. Martin (2009) menyatakan bahwa salah satu prinsip Clean Code adalah merancang kode dengan antarmuka publik yang jelas serta mengurangi akses ke detail implementasi, yang merupakan contoh langsung dari enkapsulasi [1].

Meski Python mendukung enkapsulasi, terdapat beberapa tantangan dalam penerapannya:

1. Tidak ada sistem pengendalian akses yang sepenuhnya ketat. Atribut “private” dapat diakses melalui name mangling (_ClassName__atribut).
2. Banyak pengembang baru yang melewatkan praktik ini karena Python memiliki sifat yang fleksibel dan terbuka.
3. Minimnya pendidikan formal tentang keamanan kode dan prinsip desain di antara pelajar Python

Menurut analisis literatur, beberapa saran untuk meningkatkan keamanan dan modularitas melalui enkapsulasi di Python meliputi:

1. Manfaatkan konvensi __attribute untuk mengatur akses langsung ke data.
2. Gunakan @property untuk mengatur akses yang terkendali.
3. Jelaskan dengan tegas pemisahan antara atribut internal dan eksternal.
4. Catat antarmuka publik yang diizinkan.
5. Lakukan pengujian unit pada metode publik, bukan pada atribut secara langsung.

Table 1. Sintesis Temuan

Aspek	Temuan Utama
Implementasi	Python mendukung enkapsulasi melalui konvensi, bukan kontrol akses eksplisit
Keamanan	Enkapsulasi mengurangi risiko manipulasi data yang tidak diinginkan
Reusabilitas	Membantu modularitas dan pemeliharaan kode
Tantangan	Akses tetap bisa dilakukan secara paksa; penerapan masih lemah di kalangan pemula
Rekomendasi Praktis	Gunakan @property, konvensi atribut privat, serta struktur antarmuka yang jelas

Tabel 1. menyajikan rangkuman sintesis temuan penting dari hasil kajian pustaka tentang peran enkapsulasi dalam Python. Dari segi implementasi, Python memberikan fleksibilitas dengan pendekatan konvensi penamaan (naming convention) dibandingkan dengan kontrol akses yang jelas seperti pada Java atau C++. Ini menjadi keuntungan dalam aspek kemudahan belajar, namun juga menimbulkan tantangan terkait keamanan. Dalam hal keamanan, enkapsulasi dapat membatasi akses langsung ke atribut penting dalam objek, sehingga menurunkan risiko manipulasi data secara ilegal. Ini mendukung penerapan keamanan data yang baik, meskipun dalam Python kontrol itu bersifat ilusif (tidak ketat). Mengenai reusabilitas, enkapsulasi memungkinkan kode dirancang secara modular dan mudah dipelihara karena struktur internal objek tidak terlihat secara langsung. Namun, tantangan masih ada, terutama bagi pemula yang sering tidak memperhatikan prinsip ini karena kurangnya tekanan dari sistem. Oleh sebab itu, sangat krusial untuk menerapkan strategi pelaksanaan seperti pemakaian

@property, penamaan atribut dengan __, dan membatasi akses melalui interface yang telah terdokumentasi dengan baik.

KESIMPULAN

Studi ini bertujuan untuk mengeksplorasi fungsi enkapsulasi dalam bahasa pemrograman Python dan dampaknya terhadap keamanan serta kemampuan penggunaan kembali (reusabilitas) kode. Menurut hasil kajian pustaka yang telah dilakukan, disimpulkan bahwa enkapsulasi tetap berperan penting dalam Python, meskipun mekanisme yang ada tidak seketat bahasa pemrograman lainnya seperti Java atau C++. Python menerapkan enkapsulasi dengan menggunakan konvensi penamaan (_ dan __) serta fitur @property untuk mengelola akses ke atribut objek secara terkontrol.

Dalam aspek keamanan, enkapsulasi dapat mengurangi risiko manipulasi data secara langsung, mempertahankan konsistensi nilai, dan membatasi akses ke atribut internal yang sensitif. Dalam hal reusabilitas, enkapsulasi mendukung pemrograman yang modular, terstruktur, dan mudah dirawat, sehingga dapat dimanfaatkan kembali dalam proyek pengembangan perangkat lunak lainnya.

Meskipun begitu, tantangan utama dalam penerapan enkapsulasi di Python adalah lemahnya penegakan teknik terhadap batasan akses, serta kurangnya pemahaman para pemula untuk menerapkan prinsip ini dengan konsisten. Oleh sebab itu, diperlukan saran praktis seperti mengadopsi penggunaan @property, merancang antarmuka publik yang sederhana, serta memisahkan atribut internal dari akses eksternal dengan cara terstruktur.

Secara keseluruhan, penelitian ini menegaskan bahwa meskipun Python menawarkan fleksibilitas yang tinggi, prinsip enkapsulasi tetap penting dan berkontribusi signifikan terhadap keamanan serta kualitas rekayasa perangkat lunak jika diterapkan dengan baik. Penelitian berikutnya bisa mendalami lebih lanjut tentang penggabungan enkapsulasi dengan prinsip-prinsip OOP yang lain seperti pewarisan dan polimorfisme dalam kerangka keamanan perangkat lunak Python terkini.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan dukungan dalam penyusunan artikel ini. Ucapan terima kasih khusus disampaikan kepada para penulis dan pengembang yang telah menyediakan dokumentasi resmi serta artikel teknis yang menjadi sumber utama dalam kajian ini, seperti DataCamp, RealPython, dan GeeksForGeeks. Terima kasih juga kepada dosen pembimbing dan rekan-rekan yang telah memberikan masukan dan semangat dalam proses penulisan. Semoga artikel ini dapat memberikan manfaat bagi para mahasiswa dan pengembang perangkat lunak dalam memahami dan menerapkan prinsip enkapsulasi dalam bahasa pemrograman Python secara lebih baik dan terstruktur.

REFERENSI

Adlini, M. N., Dinda, A. H., Yulinda, S., Chotimah, O., & Merliyana, S. J. (2022). Metode Penelitian Kualitatif Studi Pustaka. *Edumaspul: Jurnal Pendidikan*, *6*(1), 974–980. https://doi.org/10.33487/edumaspul.v6i1.3394

Ali, H. M., Hamza, M. Y., & Rashid, T. A. (2024). Exploring Polymorphism: Flexibility and Code Reusability in Object-Oriented Programming. *Passer Journal of Basic and Applied Sciences*, *6*(December), 502–512. https://doi.org/10.24271/PSR.2024.189667

Avianti, N., Penelitian, P., Pendekatan, S., Matematika, B., Dan, A., Nasional, P., Statistika, M., Pendidikan, P. E., & Pers, R. (2010). *Sugiyono , Metode Penelitian Kombinasi (Mixed Methods), Bandung : Alfabeta , 2013*. 2013–2015.

Del Cid, P. J., Hughes, D., Ueyama, J., Michiels, S., & Joosen, W. (2009). DARMA: Adaptable

service and resource management for wireless sensor networks. *MidSens'09 - International Workshop on Middleware Tools, Services and Run-Time Support for Sensor Networks, Co-Located with the 10th ACM/IFIP/USENIX International Middleware Conference*, 1–6. https://doi.org/10.1145/1658192.1658193

Martin, R. (2008). Clean Code. In *Clean Code*.

Ruohonen, J., Hjerppe, K., & Rindell, K. (2021). A Large-Scale Security-Oriented Static Analysis of Python Packages in PyPI. *2021 18th International Conference on Privacy, Security and Trust, PST 2021, Pst*, 1–10. https://doi.org/10.1109/PST52912.2021.9647791