



Dasar Pemrograman Python

Belajar Python (Gratis!)

Python adalah bahasa pemrograman high-level yang sangat *powerful*, sintaksnya sederhana dan mudah dipelajari, juga memiliki performa yang bagus. Python memiliki komunitas yang besar, bahasa ini dipakai di berbagai platform diantaranya: web, data science, infrastructure tooling, dan lainnya.

E-book Dasar Pemrograman Python ini cocok untuk pembaca yang ingin mempelajari pemrograman python dalam kurun waktu yang relatif cepat, dan gratis. Konten pembelajaran pada ebook ini disajikan secara ringkas tidak bertele-tele tapi tetap mencakup point penting yang harus dipelajari.

Selain topik fundamental python programming, nantinya akan disediakan juga pembahasan *advance* lainnya, **stay tuned!**

Versi e-book: **v1.0.0-beta1.20230430**, dan versi **Python 3.10.6**.

E-book ini aktif dalam pengembangan, kami akan tambah terus konten-kontennya. Silakan cek di [Github repo](#) kami mengenai progress development e-book.

Download Ebook File (pdf)

Ebook ini bisa di-download dalam bentuk file, silakan gunakan link berikut:

Source Code Praktik

Source code contoh program bisa diunduh di github.com/novalagung/dasarpemrogramanpython-example. Dianjurkan untuk sekedar tidak copy-paste dari source code dalam proses belajar, usahakan tulis sendiri kode program agar cepat terbiasa dengan bahasa Rust.

Kontribusi

Ebook ini merupakan project open source, teruntuk siapapun yang ingin berkontribusi silakan langsung saja cek github.com/novalagung/dasarpemrogramanpython. Cek juga [halaman kontributor](#) untuk melihat list kontributor.

Lisensi dan Status FOSSA

Ebook Dasar Pemrograman Rust gratis untuk disebarluaskan secara bebas, baik untuk komersil maupun tidak, dengan catatan harus disertakan credit sumber aslinya (yaitu Dasar Pemrograman Rust atau novalagung) dan tidak mengubah lisensi aslinya (yaitu CC BY-SA 4.0). Lebih jelasnya silakan cek [halaman lisensi dan distribusi konten](#).

FOSSA Status

Author & Maintainer

Ebook ini dibuat oleh Noval Agung Prayogo. Untuk pertanyaan, kritik, dan saran, silakan drop email ke [\[email protected\]](#).



Author & Contributors

Ebook Dasar Pemrograman Python adalah project open source. Siapapun bebas untuk berkontribusi di sini, bisa dalam bentuk perbaikan typo, update kalimat, maupun submit tulisan baru.

Bagi teman-teman yang berminat untuk berkontribusi, silakan fork github.com/novalagung/dasarpemrogramanpython, kemudian langsung saja cek/buat issue kemudian submit relevan pull request untuk issue tersebut 😊.

Original Author

E-book ini di-inisialisasi oleh Noval Agung Prayogo.

Contributors

Berikut merupakan hall of fame kontributor yang sudah berbaik hati menyisihkan waktunya untuk membantu pengembangan e-book ini.

1. ... anda :-)



Lisensi & Distribusi Konten

Ebook Dasar Pemrograman Python gratis untuk disebarluaskan secara bebas, dengan catatan sesuai dengan aturan lisensi CC BY-SA 4.0 yang kurang lebih sebagai berikut:

- Diperbolehkan menyebar, mencetak, dan menduplikasi material dalam konten ini ke siapapun.
- Diperbolehkan memodifikasi, mengubah, atau membuat konten baru menggunakan material yang ada dalam ebook ini untuk keperluan komersil maupun tidak.

Dengan catatan:

- Harus ada credit sumber aslinya, yaitu Dasar Pemrograman Python atau novalagung
- Tidak mengubah lisensi aslinya, yaitu CC BY-SA 4.0
- Tidak ditambahi restrictions baru
- Lebih jelasnya silakan cek <https://creativecommons.org/licenses/by-sa/4.0/>.

FOSSA Status

Instalasi Python

Ada banyak cara yang bisa dipilih untuk instalasi Python, silakan pilih sesuai preferensi dan kebutuhan.

Instalasi Python

🕒 Instalasi di Windows

- Via [Microsoft Store Package](#)
- Via [Official Python installer](#)
- Via [Chocolatey package manager](#)
- Via [Windows Subsystem for Linux \(WSL\)](#)

🕒 Instalasi di MacOS

- Via [Homebrew](#)
- Via [Official Python installer](#)

🕒 Instalasi di OS lainnya

- Via package manager masing-masing sistem operasi

🕒 Instalasi via source code

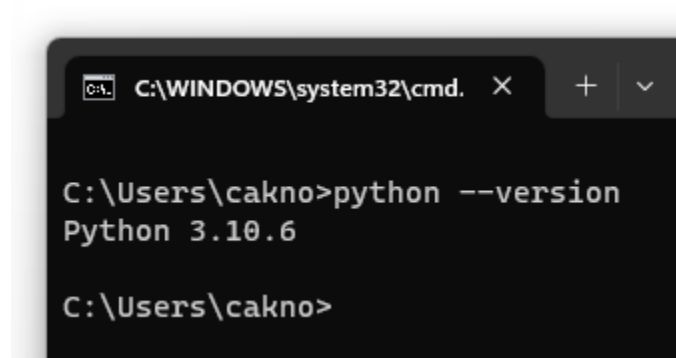
- Tarball source code bisa diunduh di [situs official Python](#)

● Instalasi via Anaconda

- File installer bisa diunduh di [situs official Anaconda](#)

Konfigurasi path Python

1. Pastikan untuk mendaftarkan path dimana Python ter-install ke OS environment variable, agar nantinya mudah dalam pemanggilan binary `python`.
2. Jika diperlukan, set juga variabel `PYTHONHOME` yang mengarah ke base folder dimana Python terinstall. Biasanya editor akan mengacu ke environment variabel ini untuk mencari dimana path Python berada.
3. Kemudian, jalankan command `python --version` untuk memastikan binary sudah terdaftar di `$PATH` variable.



```
C:\WINDOWS\system32\cmd. X + v
C:\Users\cakno>python --version
Python 3.10.6
C:\Users\cakno>
```



Python Editor & Plugin

Editor/IDE

Ada cukup banyak pilihan editor dan IDE untuk development menggunakan Python, diantaranya:

- [Eclipse](#), dengan tambahan plugin [PyDev](#)
- [GNU Emacs](#)
- [JetBrains PyCharm](#)
- [Spyder](#)
- [Sublime Text](#), dengan tambahan package [Python](#)
- [Vim](#)
- [Visual Studio](#)
- [Visual Studio Code \(VSCode\)](#), dengan tambahan extension [Python](#) dan [Jupyter](#)

Selain list di atas, ada juga editor lainnya yang bisa digunakan, contohnya seperti:

- [Python standard shell \(REPL\)](#)
- [Jupyter](#)

Preferensi editor penulis

Penulis menggunakan editor [Visual Studio Code](#) dengan tambahan:

- Extension [Python](#), untuk mendapatkan benefit API doc, autocompletion,

linting, run feature, dan lainnya.

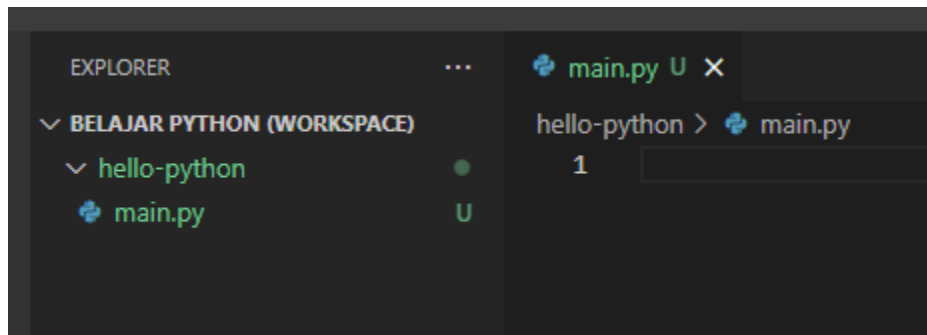
- Extension **Jupyter**, untuk interactive run program via editor.

A.1. Program Pertama → Hello Python

Bahasa pemrograman Python sangat sederhana dan mudah untuk dipelajari. Pada chapter ini kita akan langsung mempraktikannya dengan membuat program hello world.

A.1.1. Program Hello Python

Siapkan sebuah folder dengan isi satu file program Python bernama `main.py`.



Pada file `main.py`, tuliskan kode berikut:

```
print("hello python")
```

Run program menggunakan command berikut:

```
# python <nama_file_program>
```

A screenshot of a code editor interface. On the left, the 'EXPLORER' sidebar shows a workspace named 'BELAJAR PYTHON (WORKSPACE)' containing a folder 'hello-python' with a file 'main.py'. The main editor area shows the code in 'main.py':

```
hello-python > main.py
1 print("hello python")
```

Below the code editor is a 'TERMINAL' panel showing the command prompt output:

```
PS D:\Labs\hello-python> python main.py
hello python
```

Selamat, secara official sekarang anda adalah programmer Python! 🎉 Mudah bukan!?

A.1.2. Penjelasan program

Folder `hello-python` bisa disebut dengan folder *project*, dimana isinya adalah file-file program Python berekstensi `.py`.

File `main.py` adalah file program python. Nama file program bisa apa saja, tapi umumnya pada pemrograman Python, file program utama bernama `main.py`.

Command `python <nama_file_program>` digunakan untuk menjalankan program. Cukup ganti `<nama_file_program>` dengan nama file program (yang pada contoh ini adalah `main.py`) maka kode program di dalam file tersebut akan di-run oleh Python interpreter.

Statement `print("<pesan_text>")` adalah penerapan dari salah satu fungsi *built-in* yang ada dalam Python stdlib (standard library), yaitu fungsi bernama `print()` yang kegunaannya adalah untuk menampilkan pesan string (yang disiapkan di argument fungsi) ke layar output atau stdout (yang pada contoh ini adalah terminal milik editor penulis).

Lebih detailnya mengenai Python standard library (stdlib) dibahas terpisah pada chapter *Python standard library (stdlib)*

Catatan chapter

● Source code praktik

```
github.com/novalagung/dasarpemrogramanpython-example/./hello-python
```

● Referensi

- https://www.learnpython.org/en/Hello,_World!
- <https://docs.python.org/3/library/functions.html>



A.2. Run Python di VSCode

Chapter ini membahas tentang pilihan opsi cara run program Python di Visual Studio Code.


A.2.1. Cara run program Python di VSCode

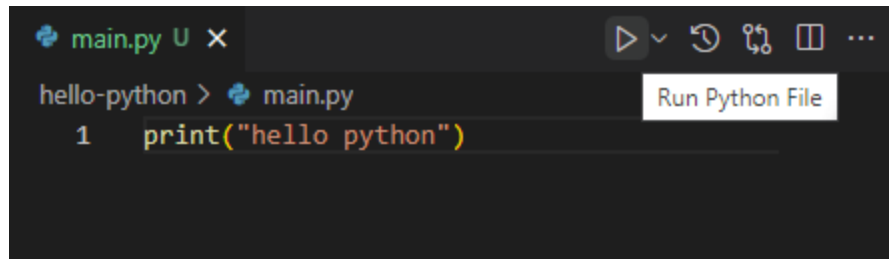
● Menggunakan command `python`

Cara ini sudah kita terapkan pada chapter [Program Pertama → Hello Python](#), dan caranya cukup mudah, tinggal jalankan saja command berikut di terminal.

```
# python <nama_file_program>  
python main.py
```

● Menggunakan tombol run

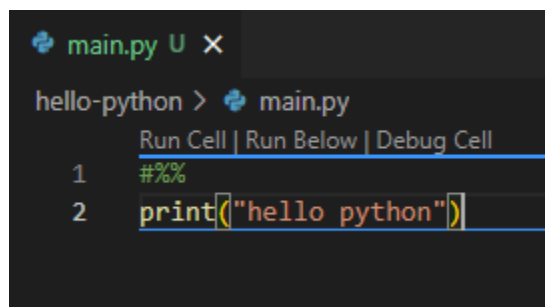
Di toolbar sebelah kanan atas ada tombol  yang bisa digunakan untuk run program.



```
main.py U x
hello-python > main.py
1 print("hello python")
```

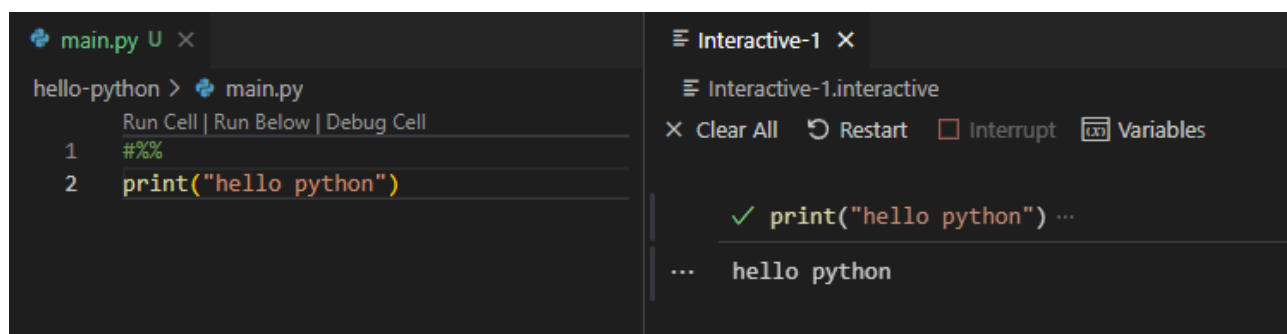
● Menggunakan jupyter code cells

Pertama tambahkan kode `#%%` pada baris di atas statement `print("hello python")`, dengan ini maka blok kode dianggap sebagai `code cell`.



```
main.py U x
hello-python > main.py
Run Cell | Run Below | Debug Cell
1 %%
2 print("hello python")
```

Setelah itu, muncul tombol `Run Cell`, klik untuk run program.



```
main.py U x
hello-python > main.py
Run Cell | Run Below | Debug Cell
1 %%
2 print("hello python")
```

Interactive-1 x

Interactive-1.interactive

Clear All Restart Interrupt Variables

✓ print("hello python") ...

... hello python

Catatan chapter

● Chapter relevan lainnya

- Program Pertama → Hello Python

● Referensi

- <https://code.visualstudio.com/docs/python/python-tutorial>
- <https://code.visualstudio.com/docs/datascience/jupyter-notebooks>
- <https://docs.python.org/3/using/cmdline.html>

A.3. Komentar

Komentar atau *mark* adalah sebuah statement yang tidak akan dijalankan oleh interpreter. Biasanya digunakan untuk menambahkan keterangan atau men-disable statements agar tidak dieksekusi saat run program.

Python mengenal dua jenis komentar, yaitu komentar satu baris dan multi-baris.

A.3.1. Komentar satu baris

Karakter `#` digunakan untuk menuliskan komentar, contoh:

```
# ini adalah komentar
print("halo,")
print("selamat pagi!") # ini juga komentar

# println("statement ini tidak akan dipanggil")
```

Jika di-run, outputnya:

```
✓ TERMINAL

PS D:\Labs\komentar> python .\main.py
halo,
selamat pagi!
```


A.3.2. Komentar multi-baris

Komentar multi-baris bisa diterapkan melalui dua cara:

● **Komentar menggunakan** `#`

```
# ini adalah komentar  
# ini juga komentar  
# komentar baris ke-3
```

● **Komentar menggunakan** `"""`

Tulis komentar multi baris kemudian apit menggunakan karakter `"""`. Contoh:

```
"""  
ini adalah komentar  
ini juga komentar  
komentar baris ke-3  
"""
```

Untuk penerapan `"""` pada komentar satu baris bisa dengan cara:

```
"""ini adalah komentar"""
```

Catatan chapter

● Source code praktik

github.com/novalagung/dasarpemrogramanpython-example/..#komentaran

● Referensi

- <https://docs.python-guide.org/writing/documentation/>

A.4. Variabel

A.4.1. Deklarasi variabel

Deklarasi variabel di Python cukup sederhana, tinggal tulis saja nama variabel kemudian diikuti operator *assignment* beserta nilai yang ingin dimasukkan ke variabel tersebut. Contoh:

```
nama = "noval"  
hobi = 'makan'  
umur = 18  
laki = True
```

Nilai string (`str`) bisa dituliskan dengan menggunakan literal `"` ataupun `'`

Selanjutnya, coba kita munculkan nilai ke-empat variabel di atas ke layar menggunakan statement `print`. Caranya:

```
print("==== biodata ====")  
print("nama: %s" % (nama))  
print("hobi: %s, umur: %d, laki: %r" % (hobi, umur, laki))
```

```
✓ TERMINAL

PS D:\Labs\variables> python main.py
==== biodata ====
nama: noval
hobi: makan, umur: 18, laki: True
```

● **Output formatting** `print`

Statement berikut adalah contoh cara memunculkan string ke layar output (`stdout`):

```
print("==== biodata ====")
```

Sedangkan contoh berikut adalah penerapan teknik *output formatting* untuk mem-format string ke layar output:

```
print("nama: %s" % (nama))
# output => "nama: noval"
```

Karakter `%s` disitu akan di-replace dengan nilai variabel `nama` sebelum dimunculkan. Dan `%s` disini menandakan bahwa data yang akan me-replace-nya bertipe data `string`.

Selain `%s` ada juga `%d` untuk data bertipe numerik integer, dan `%r` untuk data bertipe `bool`.

```
print("hobi: %s, umur: %d, laki: %r" % (hobi, umur, laki))
# output => "hobi: makan, umur: 18, laki: True"
```

Lebih detailnya mengenai output formatting dibahas terpisah pada chapter *String formatting*

A.4.2. Naming convention variabel

Mengacu ke dokumentasi [PEP 8 – Style Guide for Python Code](#), nama variabel dianjurkan untuk menggunakan `snake_case`.

```
pesan = 'halo, selamat pagi'  
nilai_ujian = 99.2
```

A.4.3. Operasi assignment

Penulisan statement operasi *assignment* sama seperti statement deklarasi variabel.

```
nama = "noval"  
umur = 18  
nama = "noval agung"  
umur = 21
```

A.4.4. Deklarasi variabel beserta tipe data

Tipe data variabel bisa ditentukan secara eksplisit, penulisannya bisa dilihat pada kode berikut:

```
nama: str = "noval"  
hobi: str = 'makan'  
umur: int = 18  
laki: bool = True  
nilai_ujian: float = 99.2
```

Lebih detailnya mengenai tipe data dibahas terpisah pada chapter *Tipe Data*

A.4.5. Deklarasi banyak variabel sebaris

Contoh penulisan deklarasi banyak variabel dalam satu baris bisa dilihat pada kode berikut:

```
nilai1, nilai2, nilai3, nilai4 = 24, 25, 26, 21  
nilai_rata_rata = (nilai1 + nilai2 + nilai3 + nilai4) / 4  
  
print("rata-rata nilai: %f" % (nilai_rata_rata))
```

Karakter `%f` digunakan untuk mem-format nilai `float`

Output program di atas:

```
▼ TERMINAL  
  
PS D:\Labs\variables> python main.py  
rata-rata nilai: 24.000
```

Catatan chapter

● Source code praktik

github.com/novalagung/dasarpemrogramanpython-example/..../variables

● Referensi

- https://www.w3schools.com/python/python_datatypes.asp
- <https://peps.python.org/pep-0008/>
- https://en.wikipedia.org/wiki/Snake_case
- https://www.learnpython.org/en/String_Formatting

A.5. Konstanta

A.5.1. Konstanta di Python

Deklarasi konstanta (atau sebuah nilai konstan yang tidak bisa diubah setelah didefinisikan) di Python bisa dilakukan menggunakan bantuan tipe *class* bernama `typing.Final`.

Untuk menggunakannya, `typing.Final` perlu di-import terlebih dahulu menggunakan statement `from` dan `import`.

```
from typing import Final
```

```
PI: Final = 3.14  
print("pi: %f" % (PI))
```

▼ TERMINAL

```
PS D:\Labs\variables> python main.py  
pi: 3.140000
```

● Module import

Keyword `import` digunakan untuk meng-import sesuatu, sedangkan keyword `from` digunakan untuk menentukan dari module mana sesuatu tersebut akan di-import.

Lebih detailnya mengenai `import` dan `from` dibahas terpisah pada

Statement `from typing import Final` artinya adalah meng-import tipe `Final` dari module `typing` yang dimana module ini merupakan bagian dari Python standard library (stdlib).

*Lebih detailnya mengenai Python standard library (stdlib) dibahas terpisah pada chapter *Python standard library (stdlib)**

A.5.2. Tipe class `typing.Final`

Tipe `Final` digunakan untuk menandai suatu variabel adalah tidak bisa diubah nilainya (konstanta). Cara penerapan `Final` bisa dengan dituliskan tipe data konstanta-nya secara eksplisit, atau tidak ditentukan.

```
# tipe konstanta PI tidak ditentukan secara eksplisit,  
# melainkan didapat dari tipe data nilai  
PI: Final = 3.14  
  
# tipe konstanta TOTAL_MONTH ditentukan secara eksplisit yaitu  
`int`  
TOTAL_MONTH: Final[int] = 12
```

*Lebih detailnya mengenai tipe data dibahas terpisah pada chapter *Tipe Data**

A.5.3. Naming convention

konstanta

Mengacu ke dokumentasi [PEP 8 – Style Guide for Python Code](#), nama konstanta harus dituliskan dalam huruf besar (UPPER_CASE).

Catatan chapter

🕒 Source code praktik

```
github.com/novalagung/dasarpemrogramanpython-example/./konstanta
```

🕒 Referensi

- <https://docs.python.org/3/library/typing.html#typing.Final>
- <https://peps.python.org/pep-0008/>

A.6. Tipe Data

Python mengenal cukup banyak tipe data, mulai dari yang *built-in* maupun custom type. Pada chapter ini kita akan belajar *high-level overview* mengenai tipe-tipe tersebut.

A.6.1. Tipe data numerik

Ada setidaknya 3 tipe data numerik di Python, yaitu:

Tipe data	Keterangan	Contoh
<code>int</code>	menampung bilangan bulat atau <i>integer</i>	<code>number_1 = 10000024</code>
<code>float</code>	menampung bilangan desimal atau <i>floating point</i>	<code>number_2 = 3.14</code>
<code>complex</code>	menampung nilai berisi kombinasi bilangan real dan imajiner	<code>number_3 = 120+3j</code>

Lebih detailnya mengenai tipe data numerik dibahas pada chapter *Tipe Data Numerik*

A.6.2. Tipe data `str`

Tipe string direpresentasikan oleh `str`, pembuatannya bisa menggunakan literal string yang ditandai dengan tanda awalan dan akhiran tanda `"` atau `'`.

- Menggunakan tanda petik dua (`"`)

```
# string sebaris
string_1 = "hello python"

# string multi-baris
string_2 = """Selamat
Belajar
Python"""
```

- Menggunakan tanda petik satu (`'`)

```
# string sebaris
string_3 = 'for the horde!'

# string multi-baris
string_4 = '''
Sesuk
Preiiii
'''
```

A.6.3. Tipe data `bool`

Literal untuk tipe data boolean di Python adalah `True` untuk nilai benar, dan

`False` untuk nilai salah.

```
bool_1 = True
bool_2 = False
```

A.6.4. Tipe data list

List adalah tipe data di Python untuk menampung nilai kolektif yang disimpan secara urut. Tipe ini biasa disebut sebagai **array**. Cara penerapan list adalah dengan menuliskan nilai kolektif dengan pembatas `,` dan diapit tanda `[` dan `]`.

```
# list with int as element's data type
list_1 = [2, 4, 8, 16]

# list with str as element's data type
list_2 = ["grayson", "jason", "tim", "damian"]

# list with various data type in the element
list_3 = [24, False, "Hello Python"]
```

Pengaksesan element list menggunakan notasi `list[index_number]`. Contoh:

```
list_1 = [2, 4, 8, 16]
print(list_1[2])
# output: 8
```

Lebih detailnya mengenai list dibahas pada chapter *Tipe Data List*

A.6.5. Tipe data tuple

Tuple adalah tipe data kolektif yang mirip dengan list, dengan perbedaan adalah:

- Nilai pada data list adalah bisa diubah (*mutable*), sedangkan nilai data `tuple` tidak bisa diubah (*immutable*).
- List menggunakan tanda `[` dan `]` untuk penulisan literal, sedangkan pada tuple yang digunakan adalah tanda `(` dan `)`.

```
# tuple with int as element's data type
tuple_1 = (2, 3, 4)

# tuple with str as element's data type
tuple_2 = ("numenor", "valinor")

# tuple with various data type in the element
tuple_3 = (24, False, "Hello Python")
```

Pengaksesan element tuple menggunakan notasi `tuple[index_number]`.
Contoh:

```
tuple_1 = (2, 3, 4)
print(tuple_1[2])
# output: 4
```

Lebih detailnya mengenai tuple dibahas pada chapter *Tipe Data Tuple*

A.6.6. Tipe data dictionary

Tipe data `dict` atau dictionary berguna untuk menyimpan data kolektif terstruktur berbentuk *key value*. Contoh penerapan:

```
profile_1 = {  
    "name": "Noval",  
    "is_male": False,  
    "age": 16,  
    "hobbies": ["gaming", "learning"]  
}
```

Pengaksesan property dictionary menggunakan notasi `dict[property_name]`. Contoh:

```
print("name: %s" % (profile_1["name"]))  
print("hobbies: %s" % (profile_1["hobbies"]))
```

Penulisan data dictionary diperbolehkan secara horizontal, contohnya seperti berikut:

```
profile_1 = { "name": "Noval", "hobbies": ["gaming", "learning"] }
```

Lebih detailnya mengenai dictionary dibahas pada chapter *Tipe Data Dictionary & Sets*

A.6.7. Tipe data sets

Tipe data sets adalah cara lain untuk menyimpan data kolektif. Tipe data ini memiliki beberapa kelemahan:

- Tidak bisa menyimpan informasi urutan data
- Elemen data yang sudah didefinisikan tidak bisa diubah nilainya (tapi bisa dihapus)
- Tidak bisa diakses menggunakan index (tetapi bisa menggunakan perulangan)

Contoh penerapan sets:

```
set_1 = {"pineapple", "spaghetti"}  
print(set_1)
```

Lebih detailnya mengenai sets dibahas pada chapter [Tipe Data Dictionary & Sets](#)

A.6.8. Tipe data lainnya

Selain tipe-tipe di atas ada juga beberapa tipe data lainnya, seperti frozenset, bytes, memoryview, range; dan kesemuanya akan dibahas satu per satu di chapter terpisah.

Catatan chapter

● Source code praktik

[github.com/novalagung/dasarpemrogramanpython-example/./tipe-data](https://github.com/novalagung/dasarpemrogramanpython-example/tree/master/tipe-data)

● Referensi

- <https://www.digitalocean.com/community/tutorials/python-data-types>
- <https://note.nkmk.me/en/python-int-max-value/>

A.7. Operator

Python mengenal beberapa jenis operator, dan pada chapter ini kita akan mempelajarinya.

A.7.1. Operator aritmatika

Operator	Keterangan	Contoh
binary <code>+</code>	operasi tambah	<code>num = 2 + 2</code> → hasilnya <code>num</code> nilainya <code>4</code>
unary <code>+</code>	penanda nilai positif	<code>num = +2 + 2</code> → hasilnya <code>num</code> nilainya <code>4</code>
binary <code>-</code>	operasi pengurangan	<code>num = 3 - 2</code> → hasilnya <code>num</code> nilainya <code>1</code>
unary <code>-</code>	penanda nilai negatif	<code>num = -2 + 3</code> → hasilnya <code>num</code> nilainya <code>1</code>
<code>*</code>	operasi perkalian	<code>num = 3 * 3</code> → hasilnya <code>num</code> nilainya <code>9</code>
<code>/</code>	operasi pembagian	<code>num = 8 / 2</code> → hasilnya <code>num</code> nilainya <code>4</code>

Operator	Keterangan	Contoh
//	operasi bagi dengan hasil dibulatkan ke bawah	num = 10 // 3 → hasilnya num nilainya 3
%	operasi modulo (pencarian sisa hasil bagi)	num = 7 % 4 → hasilnya num nilainya 3
**	operasi pangkat	num = 3 ** 2 → hasilnya num nilainya 9

A.7.2. Operator *assignment*

Operator assignment adalah `=`, digunakan untuk operasi assignment (atau penentuan nilai) sekaligus untuk deklarasi variabel jika variabel tersebut sebelumnya belum terdeklarasi. Contoh:

```
num_1 = 12
num_2 = 24

num_2 = 12
num_3 = num_1 + num_2

print(num_3)
# output: 24
```

A.7.3. Operator perbandingan

Operator perbandingan pasti menghasilkan nilai kebenaran `bool` dengan kemungkinan hanya dua nilai, yaitu benar (`True`) atau salah (`False`).

Python mengenal operasi perbandingan standar yang umumnya juga dipakai di bahasa lain.

Operator	Keterangan	Contoh
<code>==</code>	apakah kiri sama dengan kanan	<code>res = 4 == 5</code> → hasilnya <code>res</code> nilainya <code>False</code>
<code>!=</code>	apakah kiri tidak sama dengan kanan	<code>res = 4 != 5</code> → hasilnya <code>res</code> nilainya <code>True</code>
<code>></code>	apakah kiri lebih besar dibanding kanan	<code>res = 4 > 5</code> → hasilnya <code>res</code> nilainya <code>False</code>
<code><</code>	apakah kiri lebih kecil dibanding kanan	<code>res = 4 < 5</code> → hasilnya <code>res</code> nilainya <code>True</code>
<code>>=</code>	apakah kiri lebih besar atau sama dengan kanan	<code>res = 5 >= 5</code> → hasilnya <code>res</code> nilainya <code>True</code>
<code><=</code>	apakah kiri lebih kecil atau sama dengan kanan	<code>res = 4 <= 5</code> → hasilnya <code>res</code> nilainya <code>False</code>

A.7.4. Operator logika

Operator	Keterangan	Contoh
<code>and</code>	operasi logika AND	<code>res = (4 == 5) and (2 != 3) →</code> hasilnya <code>res</code> nilainya <code>False</code>
<code>or</code>	operasi logika OR	<code>res = (4 == 5) or (2 != 3) →</code> hasilnya <code>res</code> nilainya <code>True</code>
<code>not</code> atau <code>!</code>	operasi logika negasi (atau NOT)	<code>res = not (2 == 3) →</code> hasilnya <code>res</code> nilainya <code>True</code> <code>res = !(2 == 3) →</code> hasilnya <code>res</code> nilainya <code>True</code>

A.7.5. Operator bitwise

Operator	Keterangan	Contoh
<code>&</code>	operasi bitwise AND	<code>x & y = 0 (0000 0000)</code>
<code> </code>	operasi bitwise OR	<code>x y = 14 (0000 1110)</code>
<code>~</code>	operasi bitwise NOT	<code>~x = -11 (1111 0101)</code>
<code>^</code>	operasi bitwise XOR	<code>x ^ y = 14 (0000 1110)</code>

Operator	Keterangan	Contoh
>>	operasi bitwise right shift	x >> 2 = 2 (0000 0010)
<<	operasi bitwise left shift	x << 2 = 40 (0010 1000)

A.7.6. Operator *identity* (`is`)

Operator `is` memiliki kemiripan dengan operator logika `==`, perbedaannya pada operator `is` yang dibandingkan bukan nilai, melainkan identitas atau ID-nya.

Bisa saja ada 2 variabel bernilai sama tapi identitasnya berbeda. Contoh:

```
num_1 = 100001
num_2 = 100001

res = num_1 is num_2
print("num_1 is num_2 =", res)
print("id(num_1): %s, id(num_2): %s" % (id(num_1), id(num_2)))
```

▼ TERMINAL

```
PS D:\Labs\operator> python.exe main.py
num_1 is num_2 = True
id(num_1): 2545659797168, id(num_2): 2545659797168
```

DANGER

Di Python ada *special case* yang perlu kita ketahui perihal penerapan operator `is` untuk operasi perbandingan identitas khusus tipe data

numerik. Silakan cek <https://stackoverflow.com/a/15172182/1467988> untuk lebih jelasnya.

● Fungsi `print()` tanpa output formatting

Statement `print("num_1 is not num_2 =", res)` adalah salah satu cara untuk printing data tanpa menggunakan output formatting (seperti `%s`).

Yang terjadi pada statement tersebut adalah, semua nilai argument pemanggilan fungsi `print()` akan digabung dengan delimiter `|` kemudian ditampilkan ke layar console.

Agar lebih jelas, silakan perhatikan statement berikut, keduanya adalah menghasilkan output yang sama.

```
print("message: %s %s %s" % ("hello", "python", "learner"))  
print("message:", "hello", "python", "learner")
```

▼ TERMINAL

```
PS D:\Labs\operator> python.exe main.py  
message: hello python learner  
message: hello python learner
```

● Fungsi `id()`

Digunakan untuk mengambil nilai identitas atau ID suatu data. Contoh penerapannya sangat mudah, cukup panggil fungsi dan tulis data yang ingin diambil ID-nya sebagai argument pemanggilan fungsi.

```
data_1 = "hello world"
id_data_1 = id(data_1)

print("data_1:", data_1)          # hello world
print("id_data_1:", id_data_1)    # 19441xxxxxxxx
```

Nilai kembalian fungsi `id()` bertipe numerik.

A.7.7. Operator *membership* (`in`)

Operator `in` digunakan untuk mengecek apakah suatu nilai merupakan bagian dari data kolektif atau tidak.

Operator ini bisa dipergunakan pada semua tipe data kolektif seperti dictionary, sets, tuple, dan list. Selain itu, operator `in` juga bisa digunakan pada `str` untuk pengecekan substring

```
sample_list = [2, 3, 4]
print(3 in sample_list)
# False

sample_tuple = ("hello", "rust")
print("hello" in sample_tuple)
# True

sample_dict = { "nama": "noval", "age": 12 }
print("nama" in sample_dict)
# True

sample_set = { "sesuk", "preiiii" }
print("preiiii" in sample_set)
# True
```


Operator `in` jika diterapkan pada tipe dictionary, yang di-check adalah key-nya bukan value-nya.

Catatan chapter

● Source code praktik

[github.com/novalagung/dasarpemrogramanpython-example/./operator](https://github.com/novalagung/dasarpemrogramanpython-example/blob/master/operator)

● Chapter relevan lainnya

- Variabel
- Tipe Data

● Referensi

- <https://realpython.com/python-operators-expressions/>
- <https://www.programiz.com/python-programming/operators>
- <https://stackoverflow.com/a/15172182/1467988>