

## Pratikum Modul - 2



ACC  
3/11 '25

Disusun oleh :

Nama : Imam Ardi Perdana

Nim : 24241020

Kelas : PTIA

Prodi : FSTT

Dosen Pembimbing :

Adam Bachtiar, S.Kom. ,M.MT.

PROGRAM STUDI PENDIDIKAN TEKNOLOGI  
FAKULTAS SAINS DAN ILMU TERAPAN (FSTT)  
UNIVERSITAS MANDALIKA MATARAM  
2020/2025

## Praktikum Percobaan - 1

```
1 # Praktikum Percobaan - 1
2 class Hero:
3
4     jumlah = 0
5     __privateJumlah = 0
6
7     def __init__(self, name, health):
8         self.name = name
9         self.health = health
10
11     self.__private = 'private'
12     self.__protected = 'protected'
13
14 hero_1 = Hero('Atsu', 100)
15
16 print(hero_1.__dict__)
17 print(Hero.__dict__)
18 print(Hero.__privateJumlah)
```

### Outpunya:

```
{'name': 'Atsu', 'health': 100, '__Hero_private': 'private', '__Hero_protected': 'protected'}
{'__module__': '__main__', '__firstlineno__': 2, 'jumlah': 0, '__Hero_privateJumlah': 0, '__init__': <function Hero.__init__ at 0x000000E0C0A0D0>, '__static_attributes__': {'__private': '__private', '__protected': '__protected', 'health': 'health', 'name': 'name'}, '__dict__': <attribute '__dict__' of 'Hero' objects>, '__weakref__': <attribute '__weakref__' of 'Hero' objects>, '__doc__': None}
Traceback (most recent call last):
File "e:\algoritma\tugas semester 3\modul 2\PROG3D\OOP\1.Encapsulasi.py", line 18, in <module>
    print(Hero.__privateJumlah)
                                         ^
AttributeError: type object 'Hero' has no attribute '__privateJumlah'. Did you mean: '__Hero_privateJumlah'?
```

### Penjelasan:

Baris 1

Ini adalah komentar

Komentar digunakan sebagai catatan dan tidak akan  
dijalankan oleh Python

Baris 2

Class Hero : Baris ini mendefinisikan sebuah class  
bernama hero

Class digunakan sebagai template untuk membuat object.

Baris 4-5

Jumlah = 0

— Private Jumlah = 0

Ini adalah variable class

- Jumlah adalah variable Public, bisa diakses  
dari tuar class

- — Private Jumlah adalah variable Private  
(hanya bisa diakses di dalam class).

Variabel class bersifat milik class, bukan milik object.

Baris 7

```
def __init__(self, name, health):
```

Ini adalah constructor

Constructor adalah fungsi yang otomatis berjalan saat  
kita membuat object dari class.

Baris 8-9

```
    self.name = name
```

```
    self.health = health
```

Di sini kita menyimpan nilai name dan health ke dalam object.  
Setiap object akan punya name dan health masing-masing.

Baris 11-12

```
    self.__private = 'Private'
```

```
    self._protected = 'Protected'
```

- — Private = atribut Private milik object

- \_ Protected = atribut Protected (sebagai konvensi, sifatnya  
internal).

Setiap object hero akan punya dua atribut ini

Baris 14

```
hero_1 = Hero('Alsu', 100)
```

Jika membuat sebuah object baru dari class Hero bernama  
hero\_1

Object ini punya atribut :

- name = 'Alsu'

• health = 100

Baris 16

Print (hero-1 .dict -)

menampilkan semua atribut yang dimiliki oleh object hero-1 dalam bentuk dictionary.

Baris 17

Print (hero .dict -)

menampilkan semua isi class Hero seperti variable class dan fungsi-fungsinya.

Baris 18

Print (hero - PrivateJumbiah )

Baris ini akan error.

Kenapa? Karena - PrivateJumbiah adalah atribut private.

untuk mengakses harus Pakai name mangling.

Hero - Hero - PrivateJumbiah .

Variable	Jenis	Akses	Penjelasan
Jumbiah	Public class Variable	R bisa di akses dari luar class	Variable ini dapat digunakan dan diakses oleh siapa saja yang mengakses class Hero . cocok untuk nilai yang ingin diketahui oleh program lain.
- Private Jumbiah	Private class Variable	Tidak bisa di akses langsung dari luar class	Variable ini hanya bisa digunakan di dalam class Hero . Python akan melakukan name mangling . sehingga nama aslinya berubah menjadi - Hero - PrivateJumbiah . Tujuannya untuk merindungi data agar tidak diubah sembarangan .

Variable	Jenis	Akses Hasil	Penjelasan
Print (hero -1 .dict -)	menampilkan atribut milik object hero-1	Berisi data seperti name , health , dkk	
Print (hero .dict -)	menampilkan atribut milik class Hero	object hero-1	Termasuk Variable class dan Fungsi
Print (hero - Private Jumbiah )		error	Karena Variable Private tidak bisa di akses langsung . namanya aslinya menjadi - Hero - PrivateJumbiah .

## Praktikum Percobaan – 2

```
1 # Praktikum Percobaan - 2
2 class RekeningBank:
3     def __init__(self, nama, saldo):
4         self.nama = nama      # public
5         self.__saldo = saldo   # private
6
7     # method untuk menampilkan saldo
8     def lihat_saldo(self):
9         print(f"Saldo {self.nama}: {self.__saldo}")
10
11 akun_budi = RekeningBank("Budi", 1000000)
12 akun_budi.lihat_saldo() # ✅ Bisa, karena lewat method dalam class
13
14 print(akun_budi.nama)    # ✅ Bisa, karena public
15 print(akun_budi.__saldo) # ❌ ERROR, karena saldo bersifat private
16
```

Outpunya:

```
ifier.py"
Saldo Budi: 1000000
Budi
Traceback (most recent call last):
  File "e:\algoritma\Tugas-semester-3\modul 2\PRINSIP OOP\2.akses_modifier.py", line 15, in <module>
    print(akun_budi.__saldo) # ❌ ERROR, karena saldo bersifat private
                                         ^
AttributeError: 'RekeningBank' object has no attribute '__saldo'
```

Penjelasan:

Baris 1

ini hanya judul / komentar, tidak dijalankan Python

Baris 2

class Rekening\_Bank :

kita membuat class bernama Rekening\_Bank.

class ini seperti cetakan untuk membuat objek rekening.

Baris 3

def \_\_init\_\_(self, name, saldo) :

ini adalah fungsi yang otomatis jalan saat kita buat rekening baru

Isinya buat data awal: nama Pemilik dan saldo.

Baris 4

Self.nama = nama

nama Pemilik rekening

Nama boleh di akses dari luar (Public).

Baris 5

Self.saldo = Saldo

menyimpan saldo rekening

Saldo dibuat Private, artinya tidak boleh dilihat langsung dari luar class. Tujuannya supaya lebih aman

Baris 8-9

def lihat\_Saldo(self):

Print(f"Saldo (self.nama) : (self.\_saldo")

ini fungsi untuk melihat saldo. Karena fungsi ini ada di dalam class, maka dia boleh liat saldo yang Private.

Baris 11

akun\_budi = RekeningBank("budi", 1000000)

membuat rekening baru atas nama budi dengan saldo 1.000.000

Baris 12

akun\_budi.lihat\_Saldo()

menampilkan saldo budi melalui fungsi dalam class. Bisa.

Baris 14

Print(akun\_budi.nama)

menampilkan nama Pemilik rekening (Budi)

ini bisa karena nama bersifat Public.

Baris 15

Print(akun\_budi.\_saldo)

ini error, karena saldo bersifat Private. Saldo hanya boleh diakses melalui fungsi -lihat\_Saldo(), tidak boleh langsung. Apa modifier dari atribut?

modifier	contoh	Akses	Penjelasan
Public	Self.nama	bisa dari mana saja	Atribut terbuka untuk diakses
Protected	Self._saldo	Hanya dari dalam class turunan	Bisa diakses untuk dipakai secara langsung dari luar
Private	Self.__saldo	Hanya dari dalam	Tidak bisa dari luar class

## Praktikum Percobaan – 3.1

```
1 # Praktikum Percobaan 3.1 - Class Method
2 class Mobil:
3
4     total_mobil_dibuat = 0
5
6     def __init__(self, nama):
7         self.nama = nama
8         Mobil.total_mobil_dibuat += 1
9
10    def nyalakan_mesin(self):
11        print(f"Mesin {self.nama} menyala!")
12
13    @classmethod
14    def get_total_produksi(cls):
15        print(f"Pabrik telah memproduksi {cls.total_mobil_dibuat} unit mobil.")
16
17
18 Mobil.get_total_produksi()
19
20 print("--- Membuat mobil ---")
21 avanza = Mobil("Avanza")
22 xenia = Mobil("Xenia")
23 print("-----")
24
25 Mobil.get_total_produksi()
```

Outpunya:

```
PS E:\algoritma\Tugas-semester-3\modul 2> & C:/Users/USER/AppData/Local/Programs/Python/Python313/python.exe
ethad.py"
Pabrik telah memproduksi 0 unit mobil.
--- Membuat mobil ---
-----
Pabrik telah memproduksi 2 unit mobil.
```

Penjelasan:

Buris 1

Istilah untuk judul / komentar , tidak dijalankan Python

Buris 2

Class mobil:

->Bisa di cetak/blueprint berNama mobil (karena template)

Buris 4

total\_mobil\_dibuat = 0

→ angka untuk ngitung berapa mobil yang sudah dibuat  
total (Punya class, bukan Punya mobil i suju).

Baris 6-8

def \_\_init\_\_(...):

→ Fungsi yang otomatis jaln kalo kamu bikin mobil baru

- Self.nama = nama → simpan nama mobil (ini milik tiap mobil / objek).

- mobil.total\_mobil\_dibuat += 1 → setiap ada mobil baru, total ditambah 1

Baris 10-11

nyalakan\_mesin

→ fungsi buat objek mobil. Kalo dipanggil, tampilan "mesin mobil nyala".

Baris 13-15

@classmethod ... get\_total\_Produksi

→ fungsi milik class, bukan milik satu mobil.

→ Tampilkan total mobil yang dibuat.

Baris 18

mobil.get\_total\_Produksi()

→ cek total sebelum bikin mobil (hasil awal = 0)

Baris 21-22

avanza = mobil("Avanza")

Xenia = mobil("Xenia")

→ bikin 2 mobil, total mobil sekarang jadi 2

Baris 25

mobil.get\_total\_Produksi()

→ Tampilkan total mobil setelah dibuat (hasil: 2)

Jawaban Pertanyaan

1.) Atribut class yang mana?

- total\_mobil\_dibuat

(karena dipakai bersama oleh semua mobil)

2.) Atribut objek yang mana?

- Self nama

(karena tiap mobil ~~ADA~~ Punya nama? sendiri (Avanza, Xenia)).

3.) Fungsi atribut class pada kode ini?

untuk menyilung total semua mobil yang sudah dibuat

4.) method objek ~~yang~~ yang mana & fungsiya?

- nyalakan\_mesin (self)

Dipakai oleh objek mobil, contohnya Avanza.nyalakan\_mesin().

## Praktikum Percobaan - 3.2

```
1 # Praktikum Percobaan 3.2 - Static Method
2 class Kalkulator:
3
4     def __init__(self, nilai_awal):
5         self.nilai = nilai_awal
6
7     def tambah(self, angka):
8         self.nilai += angka
9         return self.nilai
10
11    @staticmethod
12    def kali(a, b):
13        return a * b
14
15 calc = Kalkulator(10)
16 print(f"Hasil tambah: {calc.tambah(5)}")
17
18 print(f"Hasil kali: {Kalkulator.kali(5, 3)}")
```

Outpunya:

```
PS E:\algoritma\Tugas-semester-3\modul 2> & C:/Users/USER/AppData/Local/Programs/Python/Python313/python.exe
c Method.py"
Hasil tambah: 15
Hasil kali: 15
```

Penjelasan:

Baris 1

Komentar / Judul : memberi keterangan "Praktikum Percobaan 3.2 - static method". Tidak diperlukan.

Baris 2

mendefinisikan class bernama kalkulator. class: catatan untuk membuat objek kalkulator.

Baris 4

mulai fungsi -init- (constructor). Fungsi ini otomatis akan  
waktu kita buat objek baru.

Baris 5

Di dalam constructor : simpan nilai -awal ke atribut self.  
Nilai -ini menjadi nilai awal untuk objek kalkulator itu.

Baris 7

mulai definisi method tambah (self.angka). Ini adalah  
method biasa yang bekerja pada objek.

Baris 8

Di method tambahan : tambahkan angka ke self.nilai  
(menyumbuh nilai yang ada di objek).

Baris 9

method tambah mengembalikan (return) nilai terbaru  
setelah penambahan, supaya bisa dipakai atau ditampilkan

Baris 11

Dekorator @staticmethod - menandai fungsi berikutnya  
sebagai static method. Artinya fungsi itu tidak butuh  
data self atau class.

Baris 12

Definisi fungsi kali (a, b) yang merupakan static  
method. Terima dua angka sebagai input.

Baris 13

Fungsi kali mengembalikan hasil perkalian  $a * b$ . Tidak  
mengubah atau memakai data objek.

Baris 15

membuat objek calc dari class kalkulator dengan nilai  
awal 10. Sekarang ada objek kalkulator bernilai 10

Baris 16

memanggil method tambah pada objek calc dengan  
argumen 5, lalu hasilnya dicetak / ditampilkan. ini juga  
mengubah nilai di dalam calc.

Baris 18

memanggil static method kali lewat class (kalkulator.kali)  
dengan argumen 5 dan 3, lalu menampilkan  
hasilnya. Karena tidak perlu buat objek untuk ini.

## Praktikum Percobaan – 4.1

```
1 # Praktikum Percobaan 4.1. - Cara Klasik : Method get_ dan set_
2 class Siswa:
3     def __init__(self, nama, nilai):
4         self.nama = nama
5         self.__nilai = 0
6         self.set_nilai(nilai)
7
8     # GETTER: Hanya untuk 'membaca'
9     def get_nilai(self):
10        print(f"(Akses getter untuk {self.nama})")
11        return self.__nilai
12
13    # SETTER: 'Satpam' untuk 'menulis'
14    def set_nilai(self, nilai_baru):
15        print(f"(Akses setter untuk {self.nama} dengan nilai {nilai_baru})")
16        if 0 <= nilai_baru <= 100:
17            self.__nilai = nilai_baru
18            print("-> Nilai berhasil diupdate.")
19        else:
20            print(f"-> Gagal! Nilai {nilai_baru} tidak valid. Harus antara 0-100.")
21
22
23 budi = Siswa("Budi", 70)
24
25 budi.set_nilai(95)
26
27 budi.set_nilai(105)
28
29 print(f"Nilai Budi sekarang: {budi.get_nilai()}")
```

**Outpunya:**

```
PS E:\algoritma\Tugas-semester-3\modul 2> & C:/Users/USER/AppData/Local/Programs/Python/Python313/python.exe
asik Method.py"
(Akses setter untuk Budi dengan nilai 70)
-> Nilai berhasil diupdate.
(Akses setter untuk Budi dengan nilai 95)
-> Nilai berhasil diupdate.
(Akses setter untuk Budi dengan nilai 105)
-> Gagal! Nilai 105 tidak valid. Harus antara 0-100.
(Akses getter untuk Budi)
Nilai Budi sekarang: 95
```

**Penjelasan:**

Baris 1

Komentar / Judul : Cuma keterangan "Praktikum Percobaan"  
4.1". Tidak dilanjutkan

Baris 2

Class Siswa.

Membuat sebuah class bernama siswa. Class = cetakan untuk  
bikin objek siswa.

Baris 4

Self. nama = nama

Simpan nama yang diberikan ke objek. Ini atribut publik  
(bisa diakses dari luar).

Baris 5

Self. -nilai = 0

Buat atribut -nilai untuk menyimpan nilai awal yang  
0 dulu. Karena Pakai ini bersifat private (tidak boleh  
diakses langsung dari luar).

Baris 6

Self. set-nilai (nilai)

Panggil method set-nilai untuk menyatur nilai awal yang  
dilajur. ini memastikan nilai awal diperiksa validasinya  
melalui setter.

Baris 8

# GETTER : Hanya untuk 'membaca'

Komentar: menjelaskan bahwa fungsi berikut adalah getter  
(hanya membaca nilai).

Baris 9

def get-nilai (self):

method getter yang digunakan untuk mengambil /  
membaca nilai siswa.

Baris 10

Print (f"(Akses getter untuk {self.nama}))")

Saat getter dipakai. tampilkan Pesan bahwa getter  
diakses untuk nama siswa tersebut.

Baris 11

return self. -nilai

kembalikan (return) nilai private - nilai sehingga  
Pemanggil bisa melihatnya.

Baris 13

# SETTER : 'SalPari' untuk 'menulis'

Komentar : setter diibaratkan saluran yang memerlukan sebelum menulis / ubah nilai.

Baris 14

def set\_nilai (self, nilai\_baru) :

Maka method setter yang menerima nilai baru untuk disimpan.

Baris 15

Print ("F" [Akses setter untuk {self.nama} dengan nilai {nilai\_baru}]) .

Tampilan Pesan bahwa setter sedang dipanggil untuk siswa itu dengan nilai yang diminta.

Baris 16

If 0 <= nilai\_baru <= 100 :

Cek apakah nilai\_baru berada dalam rentang 0 sampai 100 (valid atau tidak).

Baris 17

self.nilai = nilai\_baru

Jika valid, simpan nilai\_baru ke atribut Private\_nilai

Baris 18

Print ("→ nilai berhasil diupdate.")

Tampilkan Pesan Sukses jika nilai berhasil diubah.

Baris 19

else :

Jika nilai tidak valid (misal 105), masuk bagian ini

Baris 20

Print ("→ Gagal ! Nilai {nilai\_baru} tidak valid. Harus antara 0-100 .")

Tampilkan Pesan Gagal dan jelaskan aturannya.

Baris 23

budi = Siswa ("budi", 70)

Buat objek budi dari class Siswa dengan nama "budi" dan nilai awal 70. constructor otomatis memanggil setter sehingga nilai 70 diperiksa dan disimpan.

Baris 25

budi.set\_nilai (95)

meminta setter untuk ubah nilai budi menjadi 95. karena 95 valid (0-100), nilai diupdate dan akan tampil Pesan Sukses

Baris 27

budi.set\_nilai (105)

mencoba ubah nilai menjadi 105. karena 105 di luar rentang, setter akan menolak dan tampil Pesan error - nilai tidak berubah.

## Praktikum Percobaan – 4.2

```
1 # Praktikum Percobaan 4.2 - Cara Pythonic Menggunakan @property
2 class Siswa:
3     def __init__(self, nama, nilai):
4         self.nama = nama
5         # Saat kita menulis 'self.nilai = nilai' di sini,
6         # ini OTOMATIS memanggil method 'setter' di bawah.
7         self.nilai = nilai
8
9     # 1. GETTER (menggunakan @property)
10    @property
11    def nilai(self):
12        print("(Memanggil getter @property)")
13        return self.__nilai
14
15    # 2. SETTER (menggunakan @<nama_getter>.setter)
16    @nilai.setter
17    def nilai(self, nilai_baru):
18        print(f"(Memanggil setter @nilai.setter dengan nilai {nilai_baru})")
19
20        # Logika validasi ('satpam') tetap di sini
21        if 0 <= nilai_baru <= 100:
22            self.__nilai = nilai_baru
23            print("-> Nilai berhasil diupdate.")
24        else:
25            print(f"-> Gagal! Nilai ({nilai_baru}) tidak valid.")
26
27    # --- Mari kita coba cara baru ---
28 susi = Siswa("Susi", 80)
29 # Output:
30 # (Memanggil setter @nilai.setter dengan nilai 80)
31 # -> Nilai berhasil diupdate.
32
33 print("-" * 20)
34
35 # A. Mengubah nilai (terlihat seperti atribut, tapi memanggil SETTER)
36 susi.nilai = 101 # Ini memanggil 'def nilai(self, 101)'
37 # Output:
38 # (Memanggil setter @nilai.setter dengan nilai 101)
39 # -> Gagal! Nilai 101 tidak valid.
40
41 print("-" * 20)
42
43 susi.nilai = 90 # Ini memanggil 'def nilai(self, 90)'
44 # Output:
45 # (Memanggil setter @nilai.setter dengan nilai 90)
46 # -> Nilai berhasil diupdate.
47
48 print("-" * 20)
49
50 # B. Membaca nilai (terlihat seperti atribut, tapi memanggil GETTER)
51 print(f"Nilai Susi sekarang: (susi.nilai)")
52 # Output:
53 # (Memanggil getter @property)
54 # Nilai Susi sekarang: 90
```

Outpunya:

```
PS E:\algoritma\Tugas-semester-3\modul 2> & C:/Users/USER/AppData/Local/Programs/Python/Python313/python.exe
thonic Menggunakan property.py"
(Memanggil setter @nilai.setter dengan nilai 80)
-> Nilai berhasil diupdate.

(Memanggil setter @nilai.setter dengan nilai 101)
-> Gagal! Nilai 101 tidak valid.

(Memanggil setter @nilai.setter dengan nilai 90)
-> Nilai berhasil diupdate.

(Memanggil getter @property)
Nilai Susi sekarang: 90
```

Penjelasan:



Dipindai dengan CamScanner

Baris 1

Komentar: ini Judul Program Praktikum tentang @ Properti.

Baris 2

membuat sebuah class bernama siswa.

Baris 3

membuat method \_\_init\_\_ (konstruktor) dengan Parameter self, nama, dan nilai.

Baris 4

menimpan nilai Parameter nama ke atribut object self.nama.

Baris 5-6

Komentar: memberi Penjelasan bahwa ketika kita manulis self.nilai = nilai, Python otomatis akan memanggil method setter (yang ada di bawah).

Baris 7

self.nilai = nilai → memanggil setter untuk menyimpan nilai awal ke atribut private.\_nilai.

Baris 10-12

Decorator @ Property → mengubah method nilai () menjadi getter, sehingga kita bisa akses susi.nilai tanpa tanda kurung.

Baris 13

Definisi getter nilai (self)

Baris 14

menampilkkan Pesan bahwa getter terpanggil.

Baris 15

Return nilai Private self.\_nilai.

Baris 16

Decorator @ nilai.setter → method ini jadi setter untuk nilai

Baris 20

Definisi setter nilai (self.nilai, baru)

Baris 21

menampilkkan Pesan bahwa setter dipanggil berserta nilai barunya.

Baris 24-27

Validasi nilai : hanya menerima angka 0-100

- jika valid → simpan nilai ke self.\_nilai (baris 25).
- jika tidak valid → tampilkan Pesan gagal (baris 27)

Baris 30

membuat object susi dengan nama "susi" dan nilai awal 80

Baris 31-32

Komentar Penjelasan output

Baris 33

Output bahwa nilai awal berhasil disimpan lewat setter.

Baris 42-43

menampilkan bahwa nilai 101 tidak valid

Baris 46

ubah nilai ke 90 -> memanggil setter lagi

Baris 50

Garis Pernisah lagi

Baris 52

Print ("Nilai susi sekarang: {susi.nilai}") -> memanggil  
getter untuk mengambil nilai.

Baris 53-54

menampilkan hasil: nilai susi sekarang adalah 90.

## Tugas Praktikum Modul 2

```
1  class Hero:
2      __jumlah = 0 # private class variable
3
4      def __init__(self, nama, health, attPower, armor):
5          # Private Base Stats
6          self.__name = nama
7          self.__healthStandar = health
8          self.__attPowerStandar = attPower
9          self.__armorStandar = armor
10
11         # Level & EXP
12         self.__level = 1
13         self.__exp = 0
14
15         # Derived Stats
16         self.__healthMax = self.__healthStandar * self.__level
17         self.__attPower = self.__attPowerStandar * self.__level
18         self.__armor = self.__armorStandar * self.__level
19         self.__health = self.__healthMax
20
21     Hero.__jumlah += 1
22
23     @property
24     def gainEXP(self):
25         pass
26
27     @gainEXP.setter
28     def gainEXP(self, addEXP):
29         self.__exp += addEXP
30
31         # Level Up logic
32         while self.__exp >= 100:
33             print(f"(self.__name) level up")
34             self.__level += 1
35             self.__exp -= 100
36
37             # Update stats sesuai level baru
38             self.__healthMax = self.__healthStandar * self.__level
39             self.__attPower = self.__attPowerStandar * self.__level
40             self.__armor = self.__armorStandar * self.__level
41
42             # Reset full HP saat naik level
43             self.__health = self.__healthMax
44
45     def attack(self, musuh):
46         # Serang musuh (logika serangan belum diminta, fokus gain EXP)
47         self.gainEXP = 50
48
49     @property
50     def info(self):
51         return f"(self.__name) level {self.__level}:\n" \
52             f"\thealth = {self.__health}/{self.__healthMax}\n" \
53             f"\tattack = {self.__attPower}\n" \
54             f"\tarmor = {self.__armor}"
55
56
57     # Test
58 slardar = Hero('slardar', 100, 5, 10)
59 axe = Hero('axe', 100, 5, 10)
60
61 print(slardar.info)
62
63 slardar.attack(axe)
64 slardar.attack(axe)
65 slardar.attack(axe)
66
67 print(slardar.info)
68
```

Outpunya:

```
PS E:\algoritma\Tugas-semester-3\modul 2> & C:/Users/USER/AppData/Local/kum Modul 2.py"
slardar level 1:
    health = 100/100
    attack = 5
    armor = 10
slardar level up
slardar level 2:
    health = 200/200
    attack = 10
    armor = 20
```

Penjelasan:

Baris 1

class Hero:

membuat class bernama Hero (cetak biru untuk membuat karakter game).

Baris 2

-jumlah = 0

membuat variable Private untuk menghitung berapa hero yang sudah dibuat.

Baris 4

def \_\_init\_\_(self.name, health, attPower, armor):

Fungsi yang otomatis jalannya saat bincin Hero. Terima nama, health, attack, dan armor.

Baris 6

self.name = name

mentimpan namahero

Baris 7

self.healthStandar = health

Simpan nilai dasar HP

Baris 8

self.attPowerStandar = attPower

Simpan nilai dasar Serangan

Baris 9

self.armorStandar = armor

Simpan nilai dasar armor

Baris 12

self.level = 1

Hero mulai dari level 1

Baris 13

self.exp = 0

Exp mulai dari 0

Baris 16

self.healthMax = ...

HP maksimal = HP dasar x level

Baris 17

self.attPower = ...

Power serangan = ~~standar~~ standar x level

Baris 18

self.armor = ...

Armor = standar x level

Baris 19

self.health = self.healthMax

HP langsung Penuh di awal

Baris 21



Hero. - Jumlah t = 1

Setiap kali buat hero baru, jumlah naik 1

Baris 23

### @Property

membuat ini bisa dipanggil seperti variable

Baris 24-25

Getter gainEXP tapi kosong (Pass). Karena kita butuh

Setter saja

Baris 27

### @GainEXP. Setter

memadai fungsi berikut untuk mengatur nilai EXP

Baris 28

Setter dimulai

Baris 29

Self.EXP t = addEXP

Tambah EXP

Baris 32

while Self.EXP >= 100 :

Kalau EXP ≥ 100, naik level

Baris 33

Print hero naik level

Baris 34

level bertambah 1

Baris 35

Kurangi 100 EXP setelah naik level

Baris 38-40

Hitung ulang HP max, attack, dan armor sesuai level baru

Baris 43

Self.health = Self.health max

isi HP jadi penuh lagi setelah naik level

Baris 45 def attack (Self.musuh)

Fungsi hero menyerang (musuh belum dipukul)

Baris 47. Self.gainEXP = 50. Setiap serang dapat 50 EXP

Baris 49-51. Property info: menambahkan nama, level,

HP, attack, armor.

Baris 58

Stardar = Hero (...)

Buat hero bernama Stardar

Baris 59

Axe = Hero (...)

Buat hero bernama Axe.