

Pratikum Modul - 2



Acc
f. 10/II '25

Disusun oleh :

Nama : Rushadi

Nim : 24241016

Kelas : PTI A

Prodi : FSTT

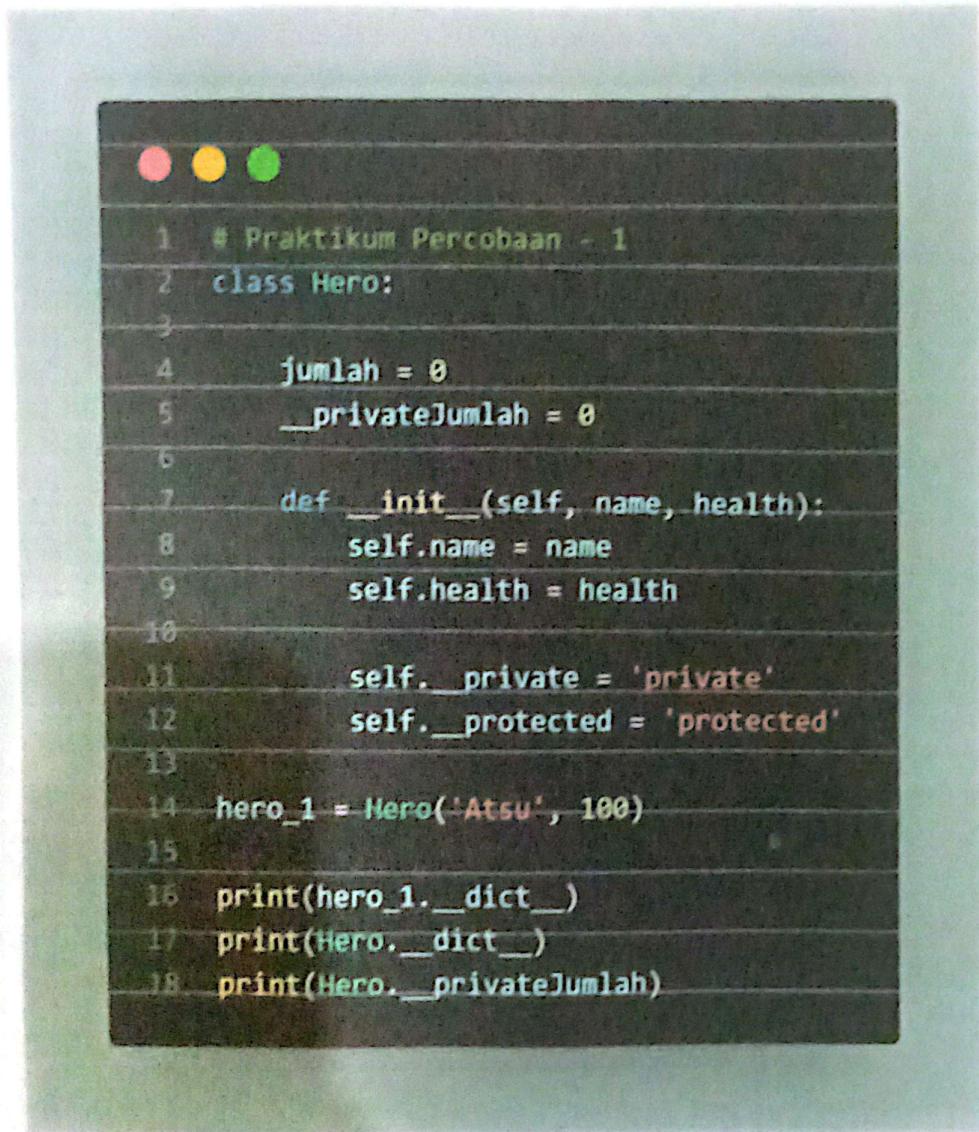
Dosen Pembimbing :

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

**PROGRAM STUDI PENDIDIKAN TEKNOLOGI
FAKULTAS SAINS DAN ILMU TERAPAN (FSTT)
UNIVERSITAS MANDALIKA MATARAM**

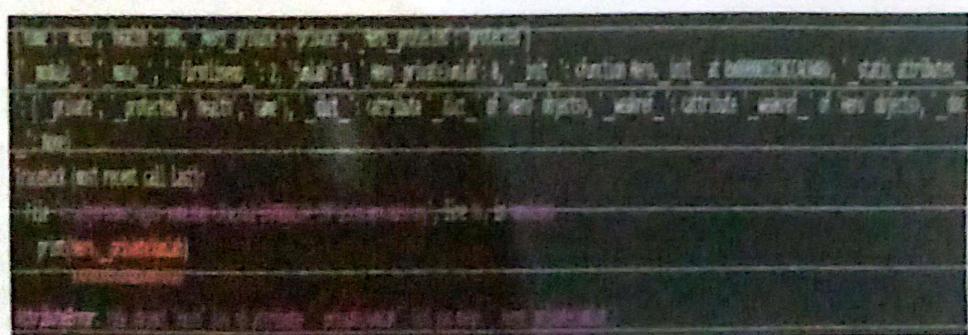
2020/2025

Praktikum Percobaan - I



```
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:



```
hero_1 = Hero('Atsu', 100)
print(hero_1.__dict__)
print(Hero.__dict__)
print(Hero.__privateJumlah)

{'name': 'Atsu', 'health': 100}
{'__privateJumlah': 0, '__private': 'private', '__protected': 'protected'}
0
```

Penjelasan:

Penjelasan :

- Baris 1 : ini adalah komentar, komentar digunakan sebagai catatan dan tidak akan dipelankan oleh Python.

- Baris 2 : Class Hero \Rightarrow Baris ini mendefinisikan sebuah class bernama Hero. Class digunakan sebagai template untuk membuat object.
- Baris 4-5 : jumlah : 0, __Privatejumlah 0 \Rightarrow ini adalah Variable Class. Jumlah adalah Variable Public, bisa diakses di luar kelas. Sedangkan __Privatejumlah adalah Variable Private (hanya bisa diakses dalam class) Variable class bersifat milik class, bukan milik object
- Baris 7 : def __init__(self, name, health) \Rightarrow ini adalah Constructor. Constructor adalah fungsi yang otomatis berjalan saat kita membuat object dari class.
- Baris 8-9 :
 - self.name = name
 - self.health = health

Disini 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 (secara konvensi, Sifatnya internal)
- Setiap object Hero akan punya dua attribute ini
- Baris 14 : hero_1 = Hero('Atsu', 100). kita membuat sebuah object baru dari class Hero bernama hero_1 object ini punya attribut:
 - name = "Atsu"
 - health = 100

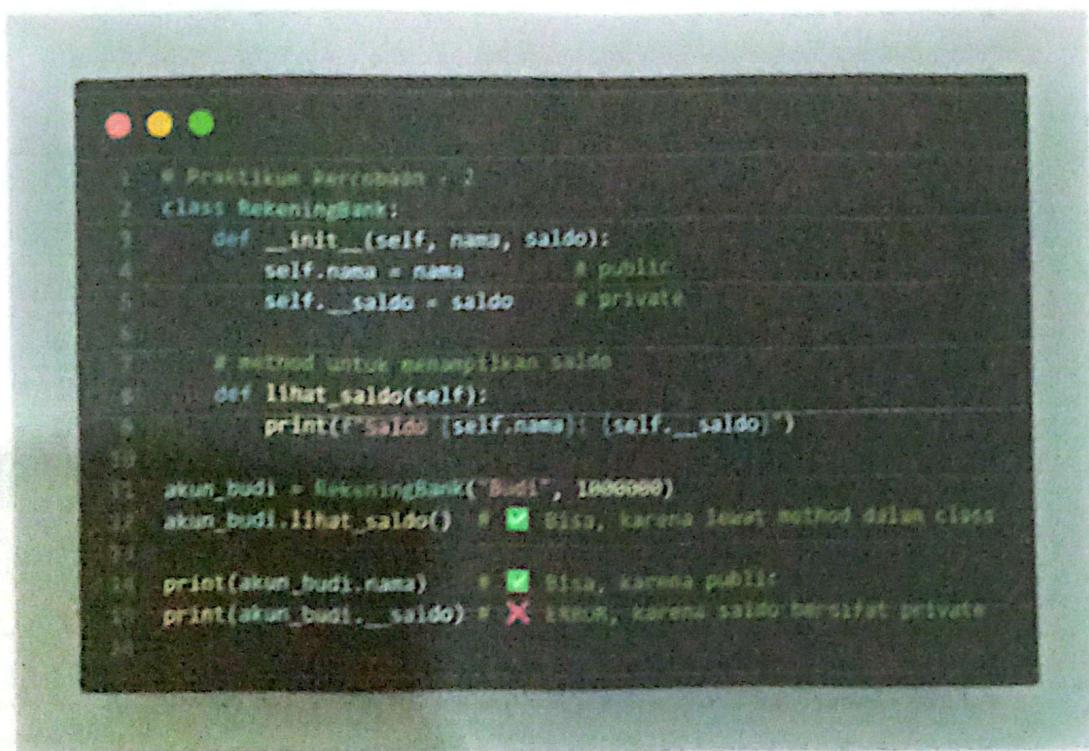
- Baris 16 : print(hero_1.__dict__)
- Menampilkan semua attribut yang dimiliki oleh object hero_1 dalam bentuk dictionary.

- Basis 17 = Print (Hero._dict_)
- Menampilkan semua isi class Hero seperti Variable class dan fungsi-fungsinya.
- Basis 18 = Print (Hero._private_jumlah_)
- Basis ini akan error. Kenapa? Karena _private_jumlah adalah atribut private. Untuk mengaksesnya harus pake name mangling : Hero._Hero._private_jumlah

2. Jelaskan Perbedaan class Variable jumlah dan __Private_jumlah

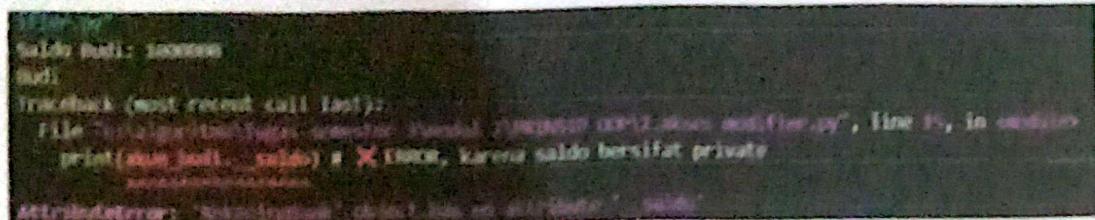
Variabel	Jenis	Akses	Pengjelasan
Jumlah	Public class Variable	Bisa diakses dari luar class	Variable ini dapat digunakan dan dilihat oleh siapa saja yang mengakses class Hero. Cade untuk nilai yang ingin diketahui oleh program lain
_private_jumlah	Private class Variable	Tidak bisa diakses langsung dari luar class	Variable ini hanya bisa digunakan di dalam class Hero. Python akan akan melakukan name mangling, sehingga nama aslinya berubah menjadi _Hero._private_jumlah. Fungsinya untuk melindungi data agar tidak diubah sembarangan
Print(hero._dict_)		Menampilkan atribut milik object hero	Perisi data seperti name, health, ds
Print(hero._dict_)		Menampilkan atribut milik class hero	Ternasuk Variable class dan fungsi
Print(Hero._private_jumlah_)		Error	Karena Variable private tidak bisa diakses langsung. Nama aslinya mengjadi _Hero._private_jumlah

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("Saldo (%s): %s" % (self.nama, self.__saldo))
10
11 akun_budi = RekeningBank("Budi", 1000000)
12 akun_budi.lihat_saldo() # ✅ bisa, karena lihat_saldo dalam class
13
14 print(akun_budi.nama)    # ✅ bisa, karena public
15 print(akun_budi.__saldo) # ❌ tidak, karena saldo bersifat private
```

Outpunya:



```
Saldo Budi: 1000000
Budi
Traceback (most recent call last):
  File "C:\Users\Budi\PycharmProjects\Python\OOP\2. class dan Function.py", line 15, in <module>
    print(akun_budi.__saldo) # ❌ tidak, karena saldo bersifat private
AttributeError: 'RekeningBank' object has no attribute '__saldo'
```

Penjelasan:

Pengmasan Praktikum 2

- Baris 1 : ini hanya judul / komentar, tidak dijalankan Python.
- Baris 2 : class Rekening_Bank ==> kita membuat class bernama RekeningBank. Class ini seperti cetakan untuk membuat objek rekening
- Baris 3 : def __init__(self, nama, saldo) : ==> ini adalah fungsi yang otomatis akan saat kita buat rekening baru. Isinya buat data awal: nama pemilik dan saldo.
- Baris 4 : self.nama = nama ==> Nama pemilik rekening. Nama tidak boleh diakses 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-g : 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 lihat saldo yang private.
- Baris 11 : akun_Budi = Rekening_Bank("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(lakun_budi._saldo) => ini error, karena saldo bersifat private. Saldo hanya boleh diakses melalui fungsi lihat_saldo(), tidak boleh langsung.
- Apa itu modififer dari atribut?

Modifier	Contoh	Akses	Pengjelasan
Public	self.nama	Bisa dari mana saja	Atribut terbuka untuk diakses
Protected	self._saldo	Hanya dari dalam class & turunan	Biasanya tidak untuk dipakai secara langsung dari luar
Private	self.__saldo	Hanya dari dalam class	Tidak bisa diakses dari luar class (dirahasiakan)

Praktikum Percobaan – 3.1

```
1 Praktikum_Percobaan_3.1 - Class Method
2
3 class Mobil:
4
5     total_mobil_dibuat = 0
6
7     def __init__(self, nama):
8         self.nama = nama
9         Mobil.total_mobil_dibuat += 1
10
11     def nyalakan_mesin(self):
12         print(f"Mesin {self.nama} menyala!")
13
14     #ClassMethod
15     @classmethod
16     def get_total_produksi(cls):
17         print(f"Pabrik telah memproduksi {cls.total_mobil_dibuat} unit mobil.")
18
19
20     Mobil.get_total_produksi()
21
22     print("... Membuat mobil ...")
23     avanza = Mobil("Avanza")
24     xenia = Mobil("Xenia")
25     print("... ")
26
27     Mobil.get_total_produksi()
```

Outpunya:

```
PS E:\algoritma\Tugas semester-3\modul-2> & c:/users/USER/AppData/Local/Programs/Python/Python33/python.exe
mobil.py
Pabrik telah memproduksi 0 unit mobil.
Membuat mobil ...
Pabrik telah memproduksi 2 unit mobil.
```

Penjelasan:

Penjelasan Praktikum 3.2

- Baris 1 : ini hanya judul / komentar, tidak digunakan Python.
- Baris 2 = class Mobil. \Rightarrow Bikin cetakan / blueprint bermacam mobil (kaya template).
- Baris 4 = total_mobil_dibuat = 0 \Rightarrow Angka untuk menghitung berapa mobil yang sudah dibuat total (Punya class, bukan punya 1 mobil saja).
- Baris 6-8: def __init__(...): \Rightarrow Fungsi otomatis jalannya kau kamu bikin mobil baru. Self.nama = nama \Rightarrow Simpan nama mobil (ini milik tiap mobil objek). Mobil.total_mobil_dibuat += 1 \Rightarrow Setiap ada mobil baru, total ditambah 1
- Baris 10-11 = nyatakan_mesin \Rightarrow fungsi buat Objek Mobil kalau dipanggil, tampilkan "mesin Mobil nya!"
- Baris 13-15 = @class method ... get_total_Produksi Fungsi milik class, bukan milik satu mobil. Tampilkan total Mobil yang udah dibuat.
- Baris 18 = Mobil.get_total_Produksi() \Rightarrow 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() \Rightarrow 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 punya nama sendiri-sendiri (Avanza, Xenia)).

3.) Fungsi Atribut class Pada kode ini?

- Untuk menghitung total semua mobil yang sudah dibuat.

4.) Method objek yang mana & fungsinya?

- nyatakan_mesin(self) (Dipakai oleh objek mobil, contohnya avanza.nyatakan_mesin()). Menampilkan tulisan kalau mesin nyala.

5.) Class Method yang mana & fungsinya?

- @ class method

- get_total_Produksi(cls)

Untuk menampilkan jumlah mobil yang sudah dibuat.

Dipanggil lewat class, bukan lewat objek.

6.) fungsi Mobil.get_total_Produksi() apa?

- Untuk mengecek dan menampilkan total mobil yang sudah dibuat.

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 C:\Users\USER\Documents\Semester 3\modul 1> & C:/Users/USER/AppData/Local/Programs/Python/Python33/python.exe
c:\Users\USER\Documents\Semester 3\modul 1>
Hasil tambah: 15
Hasil kali: 15
```

Penjelasan:

Penjelasan Praktikum 3.2

- Baris 1 = komentar /judul : memberi keterangan "Praktikum Percobaan 3.2 - static method." Tidak digunakan.
- Baris 2 = mengidentifikasi class bernama kalkulator. Class = Cetakan untuk membuat objek kalkulator.
- Baris 4 = mulai fungsi __init__ (constructor). Fungsi ini otomatis Jalan waktu kita buat objek baru.
- Baris 5 = Didalam constructor Simpan nilai_awal ke atribut self.nilai_inisiasi 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 tambah = tambahkan angka ke self.nilai (mengubah 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.
- Baris 12 = Definisi fungsi kali(a,b) yg merupakan static method. Terima dua argumen sebagai input.
- Baris 13 = fungsi kali mengembalikan hasil perkalian a,b. Tidak mengolah / memakai data objek.
- Baris 15 = Membuat calc dari class kalkulator dengan nilai awal 6. Sekarang ada nilai objek kalkulator tsb.
- Baris 16 = memanggil method tambah pada objek calc dengan argumen 5, lalu hasilnya dicetak / ditampilkan. Mengubah nilai dalam calc.
- Baris 18 = memanggil static method kali lewat class (kalkulator.kali) dengan argumen 5, 3 (itu menampilkannya langsung. Tidak perlu buat objek untuk ini)

Penjelasan perintah program

- 1. perintah print ("hasil tambah : (%s + %s)" % (calc.tambah(5), 3))
- Perintah ini memanggil method tambah(5) dari objek calc.
Objek calc memiliki nilai awal 6, kemudian method tambah (5) menambahkan angka 5 ke nilai tsb sehingga hasilnya menjadi 11. Setelah itu hasil penjumlahan tsb ditampilkan pada layar.

2. Perintah print (hasil kali: (kalkulator.kali(5,3))*)

- Perintah ini memanggil method kali(5,3) secara langsung karena nama class Kalkulator tanpa membuat objek. Method kali adalah static method, sehingga tidak membutuhkan objek untuk dijalankan. Fungsi ini hanya mengalikan angka 5x3 dan menampilkan hasilnya, yaitu 15.
- Perbedaan ke dua perintah

Perintah	cara kerja	Butuh objek atau tidak
• calc_tambah(5)	merubah nilai Objek	ya. Mengubah nilai pada objek
• kalkulator.kali(5,3)	mengalikan angka biaya	Tidak mengubah objek, hanya meng- hitung

- Perbedaan jenis method Pada Python

Jenis method	Parameter pertama	cara memanggil ketika digunakan
• interface method	self	objek.method() ketika untuk meng- akses/mengubah data pemilik
• class method	cls	classmethod() ketika operasi kelas tidak menggunakan bukan objek
• static method	Tidak ada	classmethod() cetak fungsi bantu yg tidak butuh data objek/class.

```

# membuat fungsi class Client. Method get_nam dan set_
class Siswa:
    def __init__(self, nama, nilai):
        self.nama = nama
        self.__nilai = 0
        self.set_nilai(nilai)

    # aksesor yang untuk meminta
    def get_nilai(self):
        print("akses getter untuk " + self.nama)
        return self.__nilai

    # SETTER : mengubah untuk memilih
    def set_nilai(self, nilai_baru):
        print("akses setter untuk " + self.nama + " dengan nilai " + str(nilai_baru))
        if 0 <= nilai_baru <= 100:
            self.__nilai = nilai_baru
            print("→ nilai berhasil diupdate")
        else:
            print("→ Gagal! Nilai " + str(nilai_baru) + " tidak valid. Harus antara 0-100.")

nadi = Siswa("Nadi", 70)

nadi.set_nilai(95)

nadi.set_nilai(105)

print("Nilai Nadi sekarang: " + str(nadi.get_nilai()))

```

Outpunya:

```

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

```

Penjelasan:

Penjelasan: Praktikum U.1 - cara klasik: Method get_ dan _ set

- Baris 1 = komentar / judul: cuma keterangan "Praktikum U.1" tidak dijalankan.
- Baris 2 = Data siswa, membuat sebuah class bernama siswa. class = definisi untuk buat objek siswa.
- Baris 3 = def __init__(self, nama, nilai), membuat fungsi konstruktor. Fungsi ini otomatis jalan saat kita buat objek siswa.
- Baris 4 = self.nama = nama = simpan nama yg diberikan ke objek. ini attribut objek (karena alias dari var).
- Baris 5 = self.nilai = 0, buat attribut nilai dan isi 0 data. karena pada __init__sifat private (field tidak diakses langsung)
- Baris 6 = Panggil method set_nilai untuk menghitung nilai awal di pertiksa validasinya melalui setter.
- Baris 8 = komentar : menjelaskan bahwa fungsi berikut adalah getter (hanya membaca nilai).
- Baris 9 = mulai method getter yg digunakan untuk mengambil / membaca nilai siswa.
- Baris 10 = saat getter dipakai tampilkan pesan bahwa getter diakses untuk nama siswa tsb.
- Baris 11 = kembalikan (return) nilai private — nilai sehingga memanggil bisa dilihat naja.
- Baris 13 = komentar: setter dibaratkan saat pth yg heterogen sebelum menulisubah nilai.
- Baris 14 = atau method setter gg menerima nilai baru untuk di simpan.
- Baris 15 = tampilan pesan bahwa setter sedang dipanggil untuk siswa itu dengan nilai gg dibutuhkan.

- Baris 16 = cek apakah nilai baru berada dalam rentang sampai 100 (valid/tidak).
- Baris 17 = jika valid simpan nilai baru ke atribut private_niba.
- Baris 18 = tampilkan pesan sukses jika nilai berhasil diubah.
- Baris 19 = jika nilai tidak valid (misal 105), maka bagian ini
- Baris 20 = Tampilkan pesan gagal dan jelaskan alurannya.
- Baris 23 = Buat objek budi dari class siswa dengan nama budi dan nilai awal 70. constructor otomatis memanggil setter sehingga nilai menjadi periksa dan di simpan
- Baris 25 = mencoba mengubah nilai menjadi 105, karena 105 di luar rentang, setter akan mendeklarasikan pesan error - nilai tidak berhasil.
- Baris 29 = memanggil getter untuk membaca nilai budi saat ini, lalu mengeceknya. Karena update 95 berhasil dan 105 gagal, hasil yg ditampilkan adalah 95.
- pertanyaan:

 1. Fungsi dari self.nilai = 0 dan self.set_nilai(nilai)
 - self.nilai = 0, Perintah ini untuk memberi nilai awal 0 pada atribut nilai. Jadi sebelum nilai siswa diberi, dia mulai dari angka nol dulu.
 - self.set_nilai(nilai). Perintah ini untuk memanggil fungsi setter agar nilai yg diberi saat membuat objek dicek dulu, jadi nilai awal tidak langsung dipasok, tapi dipertama dulu apakah valid(0-100) melalui setter.

2. Penjelasan baris perbaris function set_nilai()

- Def_set nilai (self, nilai_baru): merupakan method setter untuk mengubah nilai siswa.
- Print(...), menampilkan informasi bahwa kita harus mencoba mengubah nilai siswa
- if 0 < nilai_baru <= 100, memeriksa dulu apakah nilai_baru valid, yaitu berada antara 0 dan 100.
- self._nilai = nilai_baru, kalau valid nilai siswa di perbarui (di simpan ke attribut private._nilai)
- Print("Nilai berhasil diupdate") memberi pesan bahwa perubahan nilai berhasil
- else, Bagian ini jalan jika nilai_baru tidak valid.
- Print ("Gagal"), memberi pesan bahwa gagal update karena nilai tidak sesuai aturan.

3. Apa kelebihan menggunakan cara ini?

- Tidak praktis, karena harus memanggil fungsi set_nilai() sepanjang setiap kali mengubah nilai.
- Lupa menggunakan setter, sehingga ada resiko nilai berubah tanpa pengetahuan (misal orang lain melakukan akses langsung).
- Kode lebih panjang dan terlalu lelah untuk ditulis dan dilewati karena modern python yg menggunakan property, setter.

Praktikum Percobaan – 4.2

```
# praktikum_percobaan_4_2 - Cara Membuat Menggunakan property
class Siswa:
    def __init__(self, nama, nilai):
        self.nama = nama
        if 0 <= nilai <= 100:
            self.nilai = nilai
        else:
            print("Nilai harusnya berada antara 0 dan 100")
            self.nilai = nilai

    # 1. GETTER (menggunakan property)
    @property
    def nilai(self):
        print("Memanggil getter property")
        return self._nilai

    # 2. SETTER (menggunakan @name.setter)
    @nilai.setter
    def nilai(self, nilai_baru):
        print("Memanggil setter nilai.setter dengan nilai (%d)" % nilai_baru)
        if 0 <= nilai_baru <= 100:
            self._nilai = nilai_baru
            print("Nilai berhasil diupdate")
        else:
            print("=> nilai (%d) tidak valid" % nilai_baru)

    # 3. Dapat nilai maksimal dari
    susi = Siswa("Susi", 80)
    print(susi)
    # Memanggil setter susi.nilai dengan nilai 80
    # nilai berhasil diupdate
    print(susi)

    # 4. Mengambil nilai (terdapat error karena tidak menggunakan SETTER)
    susi.nilai = 90 # ini mengakibatkan error
    print(susi)
    # Memanggil setter susi.nilai dengan nilai 101
    # nilai berhasil diupdate

    print(susi)
    susi.nilai = 90 # ini mengakibatkan error
    print(susi)
    # Memanggil setter susi.nilai dengan nilai 90
    # nilai berhasil diupdate

    print(susi)
    # 5. Mengambil nilai maksimal dengan error, tapi menggunakan GETTER
    print(susi.nilai.susilakarang) # susi.nilai
```

Outpunya:

```
* PS Entalgoritma(jugos semester-1)modul 2> C:/Users/USER/AppData/Local/Programs/Python/Python311/python.exe
  File "praktikum_percobaan_4_2.py", line 1
    (Memanggil setter nilai.setter dengan nilai 80)
    ^ SyntaxError: invalid syntax
    => Nilai berhasil diupdate.

  (Memanggil setter nilai.setter dengan nilai 101)
    => Nilai (%d) tidak valid.

  (Memanggil setter nilai.setter dengan nilai 90)
    => nilai berhasil diupdate.

  (Memanggil getter property)
  nilai Susi sekarang: 90
```

Penjelasan:

praktikum 4.2 - cara python menggunakan property

- Baris 1 = judul program praktikum tentang PROPERTY.
- Baris 2 = membuat class bernama Siswa.
- Baris 3 = membuat method __init__(Konstruktor) dengan parameter self, nama, dan nilai.
- Baris 4 = menginputkan nilai parameter nama ke attribute objek self.nama.
- Baris 5-6 = memberi penjelasan bahwa menulis self.nilai = nilai; Python akan otomatis memanggil method setter (yg ada dibawah)
- Baris 7 = self.nilai → memanggil setter untuk menginput nilai awal ke attribute private __nilai.
- Baris 10-12 = Decorator property → mengubah method nilai () menjadi getter, sehingga bisa diakses sisi nilai tanpa tanda kurung.
- Baris 13 = Definisi getter nilai(self).
- Baris 14 = menambahkan pesan bahwa getter terpanggil.
- Baris 15 = Return nilai private self.__nilai.
- Baris 19 = Decorator / nilai.setter() Method ini jadi seter untuk nilai
- Baris 20 = Definisi nilai.Setter(self, nilai_baru).
- Baris 21 = menambahkan pesan bahwa Setter di panggil beserta nilai barangnya
- Baris 24 - 27 = validasi nilai: hanya menerima sampai 100. jika valid → simpan nilai ke self.__nilai (baris 25). jika tidak valid → tampilan pesan gagal (baris 27)

- Baris 30 = membuat objek susi dengan nama "sus" dan nilai awal 30.
- Baris 31-32 = Penjelasan outputnya.
- Baris 33 = Output bahwa nilai awal berhasil di simpan lewat setter
- Baris 34 = print(zo) → Hanya menampilkan garis pemisah.
- Baris 35 = mencoba mengubah nilai menjadi (01-) ini otomatis memanggil setter.
- Baris 42-43 = Menampilkan bahwa nilai 101 tidak valid
- Baris 46 = mengubah nilai menjadi 30 → memanggil setter lagi.
- Baris 50 = Garis pemisah
- Baris 52 = Menampilkan hasil: ~~sus~~ nilai susi sekarang adalah 30.

1. Perbedaan antara cara klasik dan Pythonic.

cara klasik	cara Pythonic
Method get dan set nilai	Secara manual menggunakan property atau getter dan nilai set untuk setter
Saat memanggil nilai, harus me menggil obj. set nilai	Bisa cukup menulis obj. nilai tanpa tanda kurung

2. Keuntungan memakai cara Pythonic.

- kode ringkas dan rapih
- lebih mudah dibaca
- variabel bisa dilihatkan.

Tugas Praktikum Modul 2

```
python3.7 -m venv venv & source venv/bin/activate
cd E:\Algoritma\Tugas-Semester-3\modul-2> & C:/Users/USER/AppData/Local/Java/Modul-2.py
stardar level 1:
    health = 100/100
    attack = 5
    armor = 10
stardar level up
stardar level 2:
    health = 200/200
    attack = 10
    armor = 20
print(stardar - INFO)
stardar: attack=5
stardar: attack=10
stardar: attack=20
print(stardar - INFO)
```

Outpunya:

```
ps E:\Algoritma\Tugas-Semester-3\modul-2> & C:/Users/USER/AppData/Local/Java/Modul-2.py
stardar level 1:
    health = 100/100
    attack = 5
    armor = 10
stardar level up
stardar level 2:
    health = 200/200
    attack = 10
    armor = 20
print(stardar - INFO)
stardar: attack=5
stardar: attack=10
stardar: attack=20
print(stardar - INFO)
```

Penjelasan:

Tugas Praktikum Modul 2

- Baris 1 = membuat class bernama hero (classe berfungsi membuat game)
- Baris 2 = membuat variabel private untuk menghitung berapa hero yg sudah dibuat
- Baris 4 = fungsi yg otomatis jalan saat buat hero. Terima nama, health, attack, dan armor.
- Baris 6 = menyimpan namahero
- Baris 7 = simpan nilai dasar hp.
- Baris 8 = simpan nilai dasar serangan
- Baris 9 = simpan nilai dasar armor
- Baris 12 = Hero mulai dari level 1
- Baris 13 = EXPmulai 0
- Baris 16 = HP maksimal = HP dasar x level
- Baris 17 = power serangan = standar x level
- Baris 18 = armor = standar x level
- Baris 19 = HP langsung penuh diawal
- Baris 21 = setiap kali buat hero baris jumlah naik 1
- Baris 23 = membuat property bisa dipanggil se pertama variable
- Baris 24 - 25: Getter exp jika kosong (Pass), karena belum settor saja
- Baris 27 = menandai fungsi berikan untuk mengatur nilai exp.

- Baris 28 = Softer dimuncul
- Baris 29 = Tambah EXP
- Baris 32 = Kalau EXP > 100, naik level
- Baris 33 = Print hero naik level
- Baris 34 = level bertambah
- Baris 35 = Kurangi EXP 100 setelah naik level
- Baris 38 - 40 = Hitung ulang HP maks, attack dan armor sesuai level lalu
- Baris 41 = Fungsi hero menyerang (masuk dalam dipakai)
- Baris 43 = Isi HP jadi Penuh lagi Setelah naik level
- Baris 47 = Setiap setang dapat 50 EXP.
- Baris 49 - 51 = Property info, armor/pilkkan nahas, level HP, attack, armr.
- Baris 58 = Buat hero bernama scardar
- Baris 59 = Buat hero bernama Axe.
- Baris 61 = Tampilkan status scardar aada
- Baris 63 - 65 = Standard menyerang 3 kali → fatal EXP 150 → naik level 1 (Sisa EXP 50)
- Baris 67 = Tampilkan Status scardar setelah naik level.