Pbo

4 prinsip dasar OOP ?

1. Enkapsulasi
2. Abstraksi
3. Inherintance (turunan)
4. Polymorphisme

Apa yang di maksud dengan Modul dan Package ?

* Modul :

Apa itu module pada python? Sederhananya, module pada python adalah sebuah file berekstensi .py yang berisi skrip python. Nama dari modul adalah nama nama dari file itu sendiri [2]. Misal kita memiliki file bernama 'matematika.py', maka kita telah membuat sebuah modul bernama 'matematika'.

* Paket :

Paket pada python adalah sebuah cara untuk mengelola dan mengorganisir modul- modul python dalam bentuk direktori, memungkinkan sebuah modul untuk diakses menggunakan "namespace" dan dot lokasi.

Apa singkatan dari ABC ?

In Python, abstract base classes provide a blueprint for concrete classes. They don’t contain implementation. Instead, they provide an interface and make sure that derived concrete classes are properly implemented.

* Abstract base classes cannot be instantiated. Instead, they are inherited and extended by the concrete subclasses.
* Subclasses derived from a specific abstract base class must implement the methods and properties provided in that abstract base class. Otherwise, an error is raised during the object instantiation.

Apa yang di maksud dengan Overloading ?

Overloading adalah sebuah metode yang bisa membuat dua atu lebih metode dengan nama yang sama dalam sebuah kelas, namun tipe dan jumlah argumennya harus berbeda satu sama lain.

Apa yang dimaksud dengan Agregasi ?

Agregasi adalah bentuk hubungan yang lebih khusus dari Assosiasi dimana sebuah object memiliki lifecycle nya sendiri tapi dengan kepemilikan dan class child tidak dapat memiliki class parentnya. Relasinya biasa di sebut relasi "has-a".

Jenis – jenis pewarisan :

a. Pewarisan Tunggal (Single Inheritance)

Contoh :

class Induk: # mendefinisikan kelas Induk

parent\_attr = 100

def \_\_init\_\_(self):

print ("Memanggil konstruktor induk")

def parent\_method(self):

print ('Memanggil metode induk')

def set\_attr(self, attr):

Induk.parent\_attr = attr

def get\_attr(self):

print ("Attribut induk :", Induk.parent\_attr)

class Anak(Induk): # mendefinisikan kelas Anak

def \_\_init\_\_(self):

print ("Memanggil konstruktor Anak")

def child\_method(self):

print ('Memanggil metode Anak')

c = Anak() # instansiasi kelas Anak

c.child\_method() # Anak memanggil metodenya

b. Pewarisan Ganda (Multiple Inheritance)

Contoh :

class Induk: # mendefinisikan kelas Induk

parent\_attr = 100

def \_\_init\_\_(self):

print ("Memanggil konstruktor induk")

def parent\_method(self):

print ('Memanggil metode induk')

def set\_attr(self, attr):

Induk.parent\_attr = attr

def get\_attr(self):

print ("Attribut induk :", Induk.parent\_attr)

class Anak1(Induk): # mendefinisikan kelas Anak

def \_\_init\_\_(self):

print ("Memanggil konstruktor Anak")

def child\_method(self):

print ('Memanggil metode Anak')

class Anak2(Induk, Anak1):

def \_\_init\_\_(self):

print ("Memanggil konstruktor Anak")

def child\_method(self):

print ('Memanggil metode Anak')

c. Pewarisan Multilevel (Multilevel Inheritance)

Contoh :

class Induk: # mendefinisikan kelas Induk

parent\_attr = 100

def \_\_init\_\_(self):

print ("Memanggil konstruktor induk")

def parent\_method(self):

print ('Memanggil metode induk')

def set\_attr(self, attr):

Induk.parent\_attr = attr

def get\_attr(self):

print ("Attribut induk :", Induk.parent\_attr)

class Anak1(Induk): # mendefinisikan kelas Anak

def \_\_init\_\_(self):

print ("Memanggil konstruktor Anak")

def child\_method(self):

print ('Memanggil metode Anak')

class Anak2(Anak1):

def \_\_init\_\_(self):

print ("Memanggil konstruktor Anak")

def child\_method(self):

print ('Memanggil metode Anak')

d. Pewarisan secara Hirarki (Hierarchical Inherintance)

Contoh :

class Induk: # mendefinisikan kelas Induk

parent\_attr = 100

def \_\_init\_\_(self):

print ("Memanggil konstruktor induk")

def parent\_method(self):

print ('Memanggil metode induk')

def set\_attr(self, attr):

Induk.parent\_attr = attr

def get\_attr(self):

print ("Attribut induk :", Induk.parent\_attr)

class Anak1(Induk): # mendefinisikan kelas Anak

def \_\_init\_\_(self):

print ("Memanggil konstruktor Anak")

def child\_method(self):

print ('Memanggil metode Anak')

class Anak2(Induk):

def \_\_init\_\_(self):

print ("Memanggil konstruktor Anak")

def child\_method(self):

print ('Memanggil metode Anak')

e. Pewarisan secara Hybrid (Hybrid Inheritance)

Contoh :

class Induk: # mendefinisikan kelas Induk

parent\_attr = 100

def \_\_init\_\_(self):

print ("Memanggil konstruktor induk")

def parent\_method(self):

print ('Memanggil metode induk')

def set\_attr(self, attr):

Induk.parent\_attr = attr

def get\_attr(self):

print ("Attribut induk :", Induk.parent\_attr)

class Anak1(Induk): # mendefinisikan kelas Anak

def \_\_init\_\_(self):

print ("Memanggil konstruktor Anak")

def child\_method(self):

print ('Memanggil metode Anak')

class Anak2(Induk, Anak1):

def \_\_init\_\_(self):

print ("Memanggil konstruktor Anak")

def child\_method(self):

print ('Memanggil metode Anak')

class Anak3(Anak1, Anak2):

def \_\_init\_\_(self):

print ("Memanggil konstruktor Anak")

def child\_method(self):

print ('Memanggil metode Anak')