MODUL PRAKTIKUM

PEMROGRAMAN LANJUT

AMAK YUNUS E.P

Modul Pemrograman Lanjut python

Modul 1

Dasar class

Untuk mendefinisikan kelas dengan Python, Anda dapat menggunakan kata kunci **class** , diikuti dengan nama kelas dan titik dua.

```
# deklarasi class
# class [nama class]:

class manusia:

    # Atribut.
    # Atribut ini merupakan bagian yang berisi data yang
    # memiliki tipe data tertentu
    # perhatikan contoh di bawah ini:

    nama="Wira" # atribut 1 namanya Wira
    tinggi=170  # atribut 2 tinggi 170 cm
    makan=3     # atribut 3 makan 3 kali
    lari=60     # atribut 4 kecepatan lari 60 km/jam

    # Method
    # Method merupakan fungsi yang berada dalam class.
    # method di dalam class minimal harus memiliki
    # satu parameter yaitu self
    # sintaks penulisan method adalah:
    #     def [nama method](self):
    # perhatikan contoh di bawah ini
```

```
    def punya_tinggi_badan(self):
        print("aku punya tinggi badan")

    def tinggi_badanku_adalah(self):
        return self.tinggi

# membuat instance objet
# sintaks membuat object adalah
# [nama object]=[nama class]()
# perhatikan contoh berikut

wira=manusia()

# cara memanggil/menggunakan method di dalam python
# sintaks menggunakan method
# adalah: [nama object].[nama method]()


wira.punya_tinggi_badan()

print("tinggi badan:",wira.tinggi_badanku_adalah())
```

Hasilnya adalah:

```
aku punya tinggi badan
tinggi badan: 170
```

tugas

1. Buatlah method yang lain berdasarkan atribut yang ada

Modul Pemrograman Lanjut python

Modul 2

Method dengan Parameter

Pada bagian ini, kita akan mempelajari tentang bagaimana caranya untuk menggunakan method dengan parameter tertentu. Hal ini akan bermanfaat apabila kita ingin merubah atau memasukkan data yang merupakan atribut dari class tersebut.

```
# deklarasi class
# class [nama class]:

class manusia:

   # Atribut.

   nama="Wira" # namanya Wira
   tinggi=170  # tinggi 170 cm
   makan=3     # makan 3 kali
   lari=60     # keceptan lari 60 km/jam

   # Method dengan parameter
  # sedangkan untuk method dengan parameter memiliki sintaks
   # sebagai berikut:
   #   def [nama method](self, [parameter1],[parameter2]):
   # perhatikan contoh di bawah ini

   def punya_tinggi_badan(self, tingginya):
         self.tinggi=tingginya
      # di sini, pada method punya_tinggi_badan, ada
```

```
        # dua parameter yaitu self itu sendiri dan parameter tingginya
        # ini akan merubah tinggi sesuai dengan
        # tinggi yang dimasukkan

    def tinggi_badanku_adalah(self):
        return self.tinggi

    def ganti_nama(self, namanya):
        self.nama=namanya

    def get_nama(self):
        return self.nama



# deklarasi object pada class
# memiliki bentuk sebagai berikut
# [nama object]=[nama class]()
wira=manusia()


# cara memanggil/menggunakan method dengan parameter di dalam python
# adalah: [nama object].[nama method](input)]

# di bawah ini adalah penggunaan method dengan
# parameter yang memiliki input 180
wira.punya_tinggi_badan(180)
print("tinggi badan:",wira.tinggi_badanku_adalah())
```

```
wira.ganti_nama("venus")
print("namanya berubah menjadi:",wira.get_nama())
```

Hasilnya adalah:

aku punya tinggi badan
tinggi badan: 170

Tugas

1. Buatlah method lain dari contoh di atas
2. Buatlah class petani beserta atribut, method dan pemanggilan methodnya

Modul Pemrograman Lanjut python

Modul 3

Inheritance

Inheritance adalah kemampuan suatu kelas untuk memperoleh atau mewarisi properti dari kelas lain. Kelas yang memperoleh properti disebut kelas turunan atau kelas anak, dan kelas asal properti tersebut disebut kelas dasar atau kelas induk. Manfaat inheritance adalah:

• Ini mewakili hubungan dunia nyata dengan baik.

• Menyediakan penggunaan kembali suatu kode. Kita tidak perlu menulis kode yang sama berulang kali. Selain itu, ini memungkinkan kita menambahkan lebih banyak fitur ke kelas tanpa mengubahnya.

• Bersifat transitif, artinya jika kelas B mewarisi kelas A yang lain, maka semua subkelas B otomatis mewarisi kelas A.

Jenis Inheritance

• Inheritance Tunggal: Inheritance tingkat tunggal memungkinkan kelas turunan mewarisi karakteristik dari kelas induk tunggal.

• Inheritance Multilevel: Inheritance multi-level memungkinkan kelas turunan mewarisi properti dari kelas induk langsung yang pada gilirannya mewarisi properti dari kelas induknya.

• Inheritance Hierarki: Inheritance tingkat hierarki memungkinkan lebih dari satu kelas turunan mewarisi properti dari kelas induk.

• Inheritance Berganda: Inheritance berjenjang memungkinkan satu kelas turunan mewarisi properti dari lebih dari satu kelas dasar.

Inheritance dengan Python

Pada artikel di atas, kita telah membuat dua kelas yaitu Person (kelas induk) dan Karyawan (Kelas Anak). Kelas Employee mewarisi dari kelas Person. Kita dapat menggunakan metode kelas person melalui kelas karyawan seperti yang terlihat pada fungsi tampilan pada kode di atas. Kelas anak juga bisa mengubah perilaku kelas induk seperti yang terlihat melalui metode detail().

```
1  class Manusia(object):
2      nama=""
3      umur=0
4
5      def __init__ (self,nama2):
6          self.nama=nama2
7          print("Object manusia diciptakan...! Namanya:", self.nama)
8
9
```

```
10
11 class Tentara(Manusia):
12         Amo=0
13         Granat=0
14
15         def __init__(self,nama1,umur1,Amo1,Granat1):
16             self.nama=nama1
17             self.umur=umur1
18             self.Amo=Amo1
19             self.Granat=Granat1
20             print("Object Tentara berhasil diciptakan...!!!.
Namanya:",self.nama)
21
22             def get_nama_Tentara(self):
23              print("Nama tentara",self.nama)
24
25 putri=Manusia("putri")
26 Wati=Tentara("Wati",3,100,5)
```

Modul Praktikum 4

OOP Python

Poliformisme

Apa itu Polimorfisme?

Polimorfisme diambil dari kata Yunani Poly (banyak) dan morphism (bentuk). Artinya nama fungsi yang sama dapat digunakan untuk tipe yang berbeda. Hal ini membuat pemrograman lebih intuitif dan mudah.

Di Python, kita memiliki cara berbeda untuk mendefinisikan polimorfisme. Jadi mari kita lanjutkan dan lihat cara kerja polimorfisme dengan Python.

Polimorfisme menggunakan Python

Kelas anak mewarisi semua metode dari kelas induk. Namun, dalam beberapa situasi, metode yang diwarisi dari kelas induk tidak cocok dengan kelas anak. Dalam kasus seperti itu, Anda harus menerapkan kembali metode di kelas anak.

Ada beberapa metode berbeda untuk menggunakan polimorfisme dengan Python. Anda dapat menggunakan fungsi, metode kelas, atau objek yang berbeda untuk mendefinisikan polimorfisme. Jadi, mari kita lanjutkan dan lihat masing-masing metode ini secara mendetail.

Di bawah ini ada beberapa gambar yaitu tentara, dokter, dan petani. Masing-masing memiliki peranan bagi negara, tetapi peranan mereka berbeda-beda

Gambar	Pekerjaan	peranan
	Tentara	Membela negara
	Dokter	Mengobati warga
	Petani	Menyediakan Bahan Pangan

Biasanya untuk mengetahui penggunaan polymorfism, kita menggunakan inheritane. Perhatikan contoh source code di bawah ini:

```
1   class Kendaraan:
2
3       def __init__(self, nama, warna, harga):
4           self.nama = nama
5           self.warna = warna
6           self.harga = harga
7
8       def tampilkan(self):
9           print('Details:', self.nama, self.warna, self.harga)
10
11      def kecepatan_max(self):
12          print('Kecepatan Kendaraan max = 150 km/jam')
13
14      def change_gear(self):
15          print('Kendaraan ini bisa ganti gear sampai 6')
16
17
18  # inherit dari class Kendaraan
```

```
19  class Mobil(Kendaraan):
20      def kecepatan_max(self):
21          print('Kecepatan Kendaraan max = 240 km/jam')
22
23      def change_gear(self):
24          print('Mobil ini bisa ganti gear sampai 7')
25
26
27  # Mobil Object
28  Mobil = Mobil('Mobil x1', 'Merah', 20000)
29  Mobil.tampilkan()
30  # calls methods from Mobil class
31  Mobil.kecepatan_max()
32  Mobil.change_gear()
33
34  # Kendaraan Object
35  Kendaraan = Kendaraan('Truck x1', 'putih', 75000)
36  Kendaraan.tampilkan()
37  # calls method from a Kendaraan class
38  Kendaraan.kecepatan_max()
39  Kendaraan.change_gear()
```

Hasilnya:

	Details: Mobil x1 Merah 20000 Kecepatan Kendaraan max = 240 km/jam Mobil ini bisa ganti gear sampai 7 Details: Truck x1 putih 75000 Kecepatan Kendaraan max = 150 km/jam Mobil ini bisa ganti gear sampai 6

Perhatikan pada **class Kendaraan**, terdapat dua method yaitu **tampilkan() dan kecepatan_max()**. dan pada **class mobil yang merupakan class turunan dari class kendaraan**, juga terdapat terdapat method **tampilkan() dan kecepatan_max()**. Apabila kita membuat object dari class mobil yang merupakan turunan dari class kendaraan, maka pada saat dieksekusi, method yang terpanggil/tereksekusi adalah method **tampilkan() dan kecepatan_max() dari class Mobil**.

Tugas:

Buatlah class class Manusia yang memiliki method peranan(). Kemudian Buatlah Method peranan() pada class Tentara, dokter dan petani

Modul Praktikum 5

Pemrograman lanjut Python

Method Abstraction

class Abstrak dengan Python

class abstrak dapat dianggap sebagai cetak biru untuk class lainnya. Hal ini memungkinkan Anda untuk membuat sekumpulan metode yang harus dibuat dalam setiap class anak yang dibangun dari class abstrak.

class yang berisi satu atau lebih metode abstrak disebut class abstrak. **Metode abstrak adalah metode yang mempunyai deklarasi tetapi tidak mempunyai implementasi.**

Kami menggunakan class abstrak saat kami merancang unit fungsional besar atau saat kami ingin menyediakan antarmuka umum untuk implementasi komponen yang berbeda.

class Dasar Abstrak dengan Python

Dengan mendefinisikan class dasar abstrak, Anda dapat menentukan Antarmuka Program Aplikasi (API) umum untuk sekumpulan subclass. Kemampuan ini sangat berguna dalam situasi

di mana pihak ketiga akan menyediakan implementasi, seperti dengan plugin, namun juga dapat membantu Anda ketika bekerja dalam tim besar atau dengan basis kode besar di mana mengingat semua class sulit atau tidak. mungkin.

Bekerja pada class Python Abstrak

Secara default, Python tidak menyediakan class abstrak. Python hadir dengan modul yang menyediakan dasar untuk mendefinisikan Abstract Base Class (ABC) dan nama modul tersebut adalah ABC.

ABC bekerja dengan mendekorasi metode class dasar sebagai abstrak dan kemudian mendaftarkan class konkret sebagai implementasi dari basis abstrak. Suatu metode menjadi abstrak jika dihias dengan kata kunci @abstractmethod.

Contoh 1:

Kode ini mendefinisikan class dasar abstrak yang disebut "Poligon" menggunakan modul ABC (class Dasar Abstrak) dengan Python. class "Polygon" memiliki metode abstrak yang disebut "noofsides" yang perlu diimplementasikan oleh subclassnya.

Ada empat subclass "Poligon" yang ditentukan: "Segitiga", "Pentagon", "Hexagon", dan "Segi Empat". Masing-masing subclass ini mengesampingkan metode "noofsides" dan menyediakan implementasinya sendiri dengan mencetak jumlah sisi yang dimilikinya.

Dalam kode driver, instance dari setiap subclass dibuat, dan metode "noofsides" dipanggil pada setiap instance untuk menampilkan jumlah sisi khusus untuk bentuk tersebut.

Sintaks

```
from abc import ABC, abstractmethod

class [nama_class_abstract](ABC):
    @abstractmethod
        def [nama_method1](self):
        pass


    @abstractmethod
        def stop_engine(self):
        pass
```

1	# Program python untuk menunjukkan
2	# penggunaan class abstrat method

```
3  from abc import ABC, abstractmethod
4
5
6  class Poligon(ABC):
7
8      @abstractmethod
9      def Jumlahsisi(self):
10         pass
11
12
13 class segitiga(Poligon):
14
15     # overriding abstract method
16     def Jumlahsisi(self):
17         print("Saya memiliki 3 sisi")
18
19
20 class segilima(Poligon):
21
22     # overriding abstract method
23     def Jumlahsisi(self):
24         print("Saya memiliki 5 sisi")
```

```
25
26
27  class Segienam(Poligon):
28
29      # overriding abstract method
30      def Jumlahsisi(self):
31          print("Saya memiliki 6 sisi")
32
33
34  class Segiempat(Poligon):
35
36      # overriding abstract method
37      def Jumlahsisi(self):
38          print("Saya memiliki 4 sisi")
39
40
41  # Driver code
42  R = segitiga()
43  R.Jumlahsisi()
44
45  K = Segiempat()
46  K.Jumlahsisi()
```

47	
48	R = segilima()
49	R.Jumlahsisi()
50	
51	K = Segienam()
52	K.Jumlahsisi()
	Hasil: Saya memiliki 3 sisi Saya memiliki 4 sisi Saya memiliki 5 sisi Saya memiliki 6 sisi

Modul Praktikum 6

Pemrograman lanjut Python

interface

interface dengan Python dengan Contoh

Pada praktikum kali ini saya akan membahas interface dengan Python beserta Contohnya. Silakan baca praktikum sebelumnya di mana kita membahas class Abstrak dengan Python. Sebagai bagian dari praktikum ini, kita akan membahas petunjuk berikut yang terkait dengan interface dengan Python.

Pada praktikum kita sebelumnya, kita telah membahas bahwa class abstrak adalah class yang mungkin berisi beberapa metode abstrak dan juga metode non-abstrak. Bayangkan ada class abstrak yang hanya berisi metode abstrak dan tidak berisi metode konkrit, class tersebut disebut interface.

Oleh karena itu, interface tidak lain adalah class abstrak yang hanya berisi metode abstrak.

Poin yang Perlu Diingat:

Di python tidak ada kata kunci tersendiri untuk membuat interface. Kita dapat membuat interface dengan menggunakan class abstrak yang hanya memiliki metode abstrak.

Seperti yang kita ketahui, metode abstrak harus diimplementasikan di subclass interface (class Abstrak). Pembuatan objek tidak dimungkinkan untuk class interface (class abstrak). Kita dapat membuat objek untuk class interface anak untuk mengakses metode yang diimplementasikan.

Sintaks

```
from abc import ABC, abstractmethod
class [nama class interface](ABC):


    #semua yang berada dalam interface class,
    #harus berupa abtract method semua
    #dan hanya boleh deklarasi method saja


    @abstractmethod
    def [method1](self):
      pass
```

```
    @abstractmethod
    def [method2](self,[parameter1):
        pass
```

#sedangkan untuk class turunannya akan berupa seperti di bawah ini

```
class [turunan]([nama class interface]):


    def method1(self):
      #implementasi method1


    def method2(self, [parameter]):
      #implementasi method2
```

Contoh

```
1  from abc import ABC, abstractmethod  #ini wajib dipkai ya
2  class TtgInterfaceManusia(ABC):   #penggunaan ABC
3
4      #semua yang berada dalam interface class,
5      #harus berupa abtract method semua
6      #dan hanya boleh deklarasi method saja
7
8      @abstractmethod
9      def Berjalan(self):
10       pass
11
12     @abstractmethod
13     def Berlari(self,jarak):
14       pass
15
16 # di class atlit dan petani,
17 # semuanya mengimplementasikan method
18 # yang telah dideklarasikan pada abstract method
19 class Atlit(TtgInterfaceManusia):
20
```

```
21      def Berjalan(self):
22          print("Aku Atlit. aku mampu berjalan")
23
24      def Berlari(self, jarak):
25          print("Aku atlit. aku mampu berlari sejauh ", jarak, " km")
26
27
28  class Petani(TtgInterfaceManusia):
29
30      def Berjalan(self):
31          print("Aku Petani. aku mampu berjalan")
32
33      def Berlari(self, jarak):
34          print("Aku petani. aku mampu berlari sejauh ", jarak, " km")
35
36
37
38
39  roni=Atlit()
40  Paidi=Petani()
41
42  roni.Berjalan()
```

```
43  roni.Berlari(42)
44
45  Paidi.Berjalan()
46  Paidi.Berlari(10)
```

Tugas
1. Buatlah class interface tentang mobil dan deklarasikan abstrak method tentang fasilitas, kecepatan, harga, kekuatan 2. Terapkan method dari class interface tersebut ke dalam class turunan seperti BMW, avanza, Ferrari

Modul Praktikum 7

Pemrograman lanjut Python

Package

Di Python, modul adalah skrip Python dengan ekstensi .py dan berisi objek seperti kelas, fungsi, dll. Paket di Python memperluas konsep pendekatan modular lebih jauh. Paketnya adalah folder yang berisi satu atau lebih file modul; selain itu, file khusus "__init__.py" mungkin kosong tetapi mungkin berisi daftar paket.

Buat Paket Python

Mari kita membuat paket Python dengan nama **paketku** . Ikuti langkah-langkah yang diberikan di bawah ini -

1. Buat folder luar untuk menampung konten **paketku**. Biarkan namanya menjadi **demopaket**.
2. Di dalamnya, buat folder lain **paketku**. Ini akan menjadi paket Python yang akan kita buat. Dua modul Python **fungsiluas.py** dan **fungsimath.py** akan dibuat di dalam **paketku**.
3. Buat file "__.init__.py" kosong di dalam folder **paketku**.
4. Di dalam folder luar, nanti kita akan menyimpan skrip Python **contoh.py** untuk menguji package kita.

Struktur folder mengikuti bagan di bawah ini

Demopaket (Folder)

- contohpaket.py
- testingpaket.py
- setup.py
- **paketku (Folder)**
  - __init__.py
  - fungsiluas.py
  - fungsimath.py

Dengan menggunakan ode editor favorit kalian, buatlah file **__init__.py, fungsiluas.py, dan fungsimath.py pada folder paketku**.

File fungsimath.py

```
# fungsimath.py

def jumlah(x,y):

   nilai = x+y

   return nilai


def ratarata(x,y):

   nilai = (x+y)/2

   return nilai
```

```
def pangkat(x,y):
    nilai = x**y
```

File fungsiluas.py

```
# fungsiluas.py
def persegipanjang(w,h):
   luas = w*h
   return luas

def lingkaran(r):
   import math
   luas = math.pi*math.pow(r,2)
   return luas
```

Sekarang mari kita uji paket **paketku** dengan bantuan skrip Python di atas folder paket ini. Lihat struktur folder di atas.

File contohpaket.py

```
#contohpaket.py
```

```
from paketku.fungsiluas import persegipanjang

print ("Luas :", persegipanjang(10,20))


from paketku.fungsimath import ratarata

print ("Rata-rata:", ratarata(10,20))
```

Hasinya:

Luas : 200

Rata-rata: 15.0

Anda dapat meletakkan fungsi yang dipilih atau sumber daya lain dari paket di file "__init__.py". Mari kita masukkan kode berikut ke file __init__.py

File __init__.py

```
from .fungsiluas import persegipanjang, lingkaran

from .fungsimath import jumlah, ratarata, pangkat
```

Untuk mengimpor fungsi yang tersedia dari paket ini, simpan skrip berikut sebagai testingpaket.py, di atas folder paket seperti sebelumnya.

File testingpaket.py

```
#testingpaket.py

from paketku import pangkat, lingkaran


print ("luas lingkaran:", lingkaran(5))

print ("10 pangkat 2 :", pangkat(10,2))
```

Hasil:

luas lingkaran: 78.53981633974483

10 pangkat 2 : 100

Buatlah file setup.py untuk mempersiapkan instalasi paket yang telah kita buat

File setup.py

```
#setup.py

from setuptools import setup

setup(name='paketku',

version='1.0',

description='script untuk setup paket',

url='#',

author='goodman',
```

```
author_email='goodman@gmail.com',
license='MIT',
packages=['paketku'],
zip_safe=False)
```

Lalu masuklah ke command prompt (CMD) dan masuklah ke folder demopaket (**Peringatan: untuk path masing-masing komputer akan berbeda. Tergantung di drive dan folder mana anda menyimpan folder demopaket**).

Kemdian tekan perintah:

python3 -m pip install paketku

```
C:\Users\User\Documents\Python OOP\modul 1\demopaket>python3 -m pip install paketku
```

Kemudian masukkan perintah seperti di bawah ini

```
C:\Users\User\Documents\Python OOP\modul 1\demopaket>python
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import paketku
```

```
>>> paketku.lingkaran(5)
78.53981633974483
>>>
```

Referensi:

https://realpython.com/python3-object-oriented-programming/

https://www.datacamp.com/tutorial/python-oop-tutorial

https://www.geeksforgeeks.org/python-oops-concepts/

https://www.programiz.com/python-programming/object-oriented-programming

https://www.tutorialspoint.com/python/index.htmhttps://www.tutorialspoint.com/python/index.htm