LAPORAN PRAKTIKUM

PEMROGRAMAN BERORIENTASI OBJEK LANJUT

2023

Prepared By:

Umar Faqih

R2(B)

210511066

Tugas Praktikum 7 :

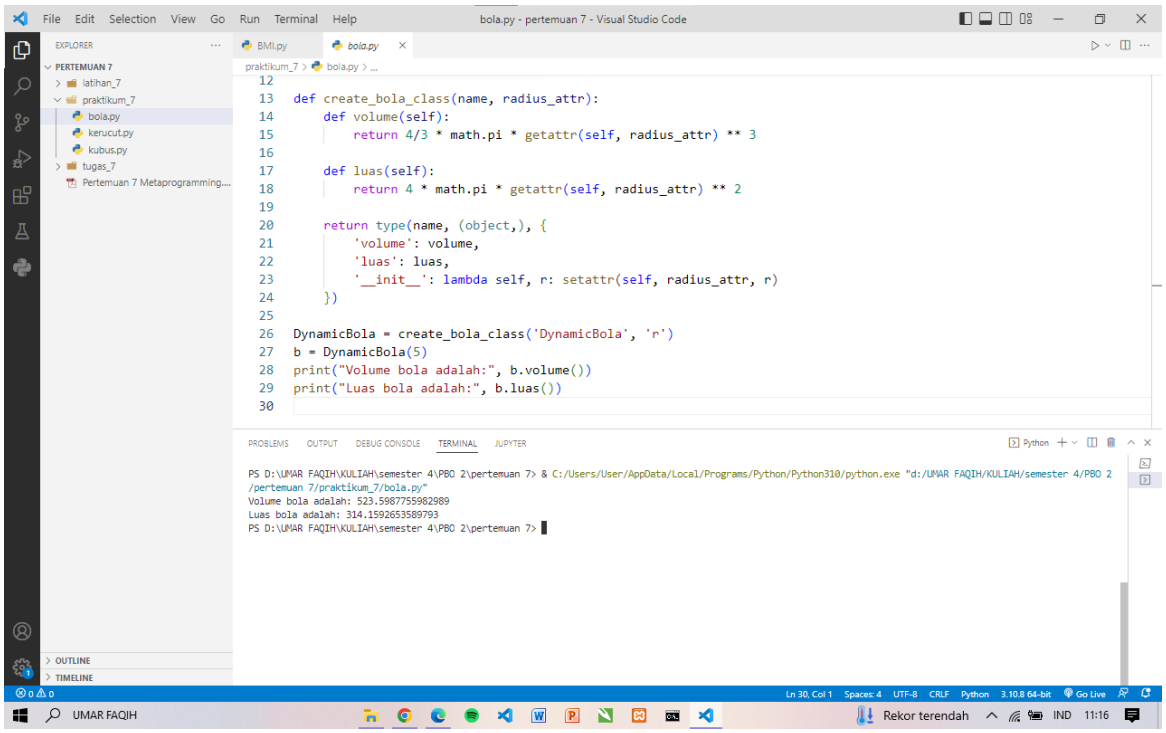
Buatlah 3 aplikasi untuk menghitung volume dan luas permukaan selain dari contoh diatas menggunakan teknik Metaprogramming. Hasilnya diupload ke github masing-masing di

1. Bola

```
import math

class Bola:
    def __init__(self, r):
        self.r = r

    def volume(self):
        return 4/3 * math.pi * self.r ** 3

    def luas(self):
        return 4 * math.pi * self.r ** 2

def create_bola_class(name, radius_attr):
    def volume(self):
        return 4/3 * math.pi * getattr(self,
radius_attr) ** 3

    def luas(self):
        return 4 * math.pi * getattr(self, radius_attr)
** 2

    return type(name, (object,), {
        'volume': volume,
        'luas': luas,
        '__init__': lambda self, r: setattr(self,
radius_attr, r)
    })

DynamicBola = create_bola_class('DynamicBola', 'r')
b = DynamicBola(5)
```

```
print("Volume bola adalah:", b.volume())
print("Luas bola adalah:", b.luas())
```

Hasil :

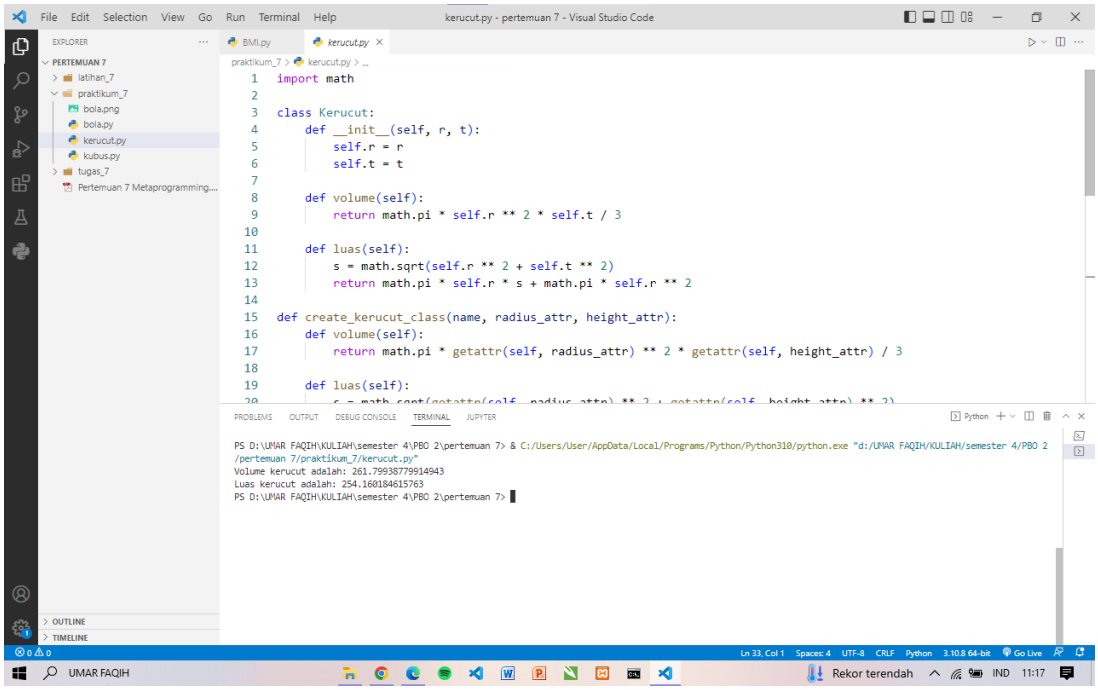


2. Kerucut

```
import math

class Kerucut:
    def __init__(self, r, t):
        self.r = r
        self.t = t

    def volume(self):
        return math.pi * self.r ** 2 * self.t / 3
```

```
    def luas(self):
        s = math.sqrt(self.r ** 2 + self.t ** 2)
        return math.pi * self.r * s + math.pi * self.r
** 2

def create_kerucut_class(name, radius_attr,
height_attr):
    def volume(self):
        return math.pi * getattr(self, radius_attr) **
2 * getattr(self, height_attr) / 3

    def luas(self):
        s = math.sqrt(getattr(self, radius_attr) ** 2 +
getattr(self, height_attr) ** 2)
        return math.pi * getattr(self, radius_attr) * s
+ math.pi * getattr(self, radius_attr) ** 2

    return type(name, (object,), {
        'volume': volume,
        'luas': luas,
        '__init__': lambda self, r, t: setattr(self,
radius_attr, r) or setattr(self, height_attr, t)
    })

DynamicKerucut = create_kerucut_class('DynamicKerucut',
'r', 't')
k = DynamicKerucut(5, 10)
print("Volume kerucut adalah:", k.volume())
print("Luas kerucut adalah:", k.luas())
```

Hasil :

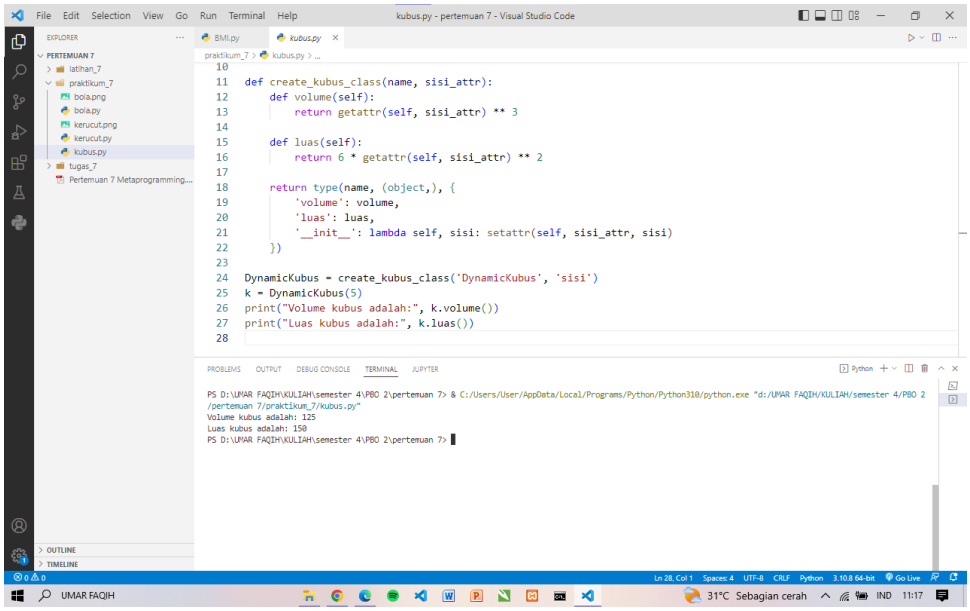


3. Kubus

```
class Kubus:
    def __init__(self, sisi):
        self.sisi = sisi

    def volume(self):
        return self.sisi ** 3

    def luas(self):
        return 6 * self.sisi ** 2

def create_kubus_class(name, sisi_attr):
    def volume(self):
        return getattr(self, sisi_attr) ** 3

    def luas(self):
```

```
        return 6 * getattr(self, sisi_attr) ** 2

    return type(name, (object,), {
        'volume': volume,
        'luas': luas,
        '__init__': lambda self, sisi: setattr(self,
sisi_attr, sisi)
    })

DynamicKubus = create_kubus_class('DynamicKubus',
'sisi')
k = DynamicKubus(5)
print("Volume kubus adalah:", k.volume())
print("Luas kubus adalah:", k.luas())

    asyncio.run(main())

except StopAsyncIteration as e:
    print("Iterasi berhenti:", e)
```

Hasil :



Link github < https://github.com/Umar-Faqih/PBO-2/tree/main/Praktikum >