

import copy

import math

class TCircle:

def \_\_init\_\_(self, \*args):

arguments\_count=len(args)

if arguments\_count==0:

self.radius=0

elif arguments\_count==1:

if type(args[0])==int or type(args[0])==float:

self.radius=args[0]

elif type(args[0])==str:

arg\_list=args[0].split(' ')

if len(arg\_list)==1:

self.radius=float(arg\_list[0])

else:

raise Exception("Неправильний аргумент!!!")

elif arguments\_count==1:

self.radius=args[0]

else:

raise Exception("Неправильний аргумент!!!")

@classmethod

def from\_tcircle(cls, class\_instance):

radius = copy.deepcopy(class\_instance.radius)

return cls(radius)

def input(self):

self.radius=int(input("Введіть радіус кола: "))

print("Ваш радіус = {0}".format(self.radius))

return self.radius

def str(self):

return str(self.radius)

def repr(self):

return self.radius

def S\_circle(self):

s=math.pi\*(self.radius)\*\*2

print("Радіус кола= {0}".format(s))

self.s=s

return s # Не зберігає s --------------------------------------

def len\_circle(self):

l=2\*math.pi\*self.radius

print("Довжина кола= {0}".format(l))

return l

def \_\_eq\_\_(self, other\_circle):

# print("eqql")

return self.s() < other\_circle.s()

def \_\_add\_\_(self, other):

return TCircle(other.radius + self.radius)

def \_\_sub\_\_(self, other):

return TCircle(math.fabs(other.radius - self.radius))

def \_\_mul\_\_(self, other):

return TCircle(other \* self.radius)

c1=TCircle(60)

c2=TCircle(90)

c1.S\_circle()

c1.len\_circle()

c3=c1+c2

print(c3.repr())

c4=c1-c2

print(c4.repr())

mu=c1\*3

mu.str()

c1=c2

if c1==c2:

print("Кола рівні")

class Cone(TCircle):

def \_\_init\_\_(self,radius,h):

super().\_\_init\_\_(self,radius)

self.h=h

def V(self):

return super().s()\*self.h \*(1/3)

con=Cone(30,6)

print("v=",con.V())