**Code:**

class Test:

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

self.x = 0

class Derived\_Test(Test):

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

self.y = 1

def main():

b = Derived\_Test()

print(b.x,b.y)

main()

**Output**

Traceback (most recent call last):

File "Task6.4.py", line 13, in <module>

main()

File "Task6.4.py", line 11, in main

print(b.x,b.y)

AttributeError: 'Derived\_Test' object has no attribute 'x'

**Code:**

class A:

def \_\_init\_\_(self, x= 1):

self.x = x

class der(A):

def \_\_init\_\_(self,y = 2):

super().\_\_init\_\_()

self.y = y

def main():

obj = der()

print(obj.x, obj.y)

main()

**Output:**

1 2

**Code:**

class A:

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

self.x = x

def count(self,x):

self.x = self.x+1

class B(A):

def \_\_init\_\_(self, y=0):

A.\_\_init\_\_(self, 3)

self.y = y

def count(self):

self.y += 1

def main():

obj = B()

obj.count()

print(obj.x, obj.y)

main()

**Output:**

3 1

**Code:**

class A:

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

self.multiply(15)

print(self.i)

def multiply(self, i):

self.i = 4 \* i;

class B(A):

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

super().\_\_init\_\_()

def multiply(self, i):

self.i = 2 \* i;

obj = B()

**Output:**

30