**Abstract class and Method Implementation**

.

1. Write a program to make the class as abstract class using ABC class and create object for the class.

from abc import ABC,abstractclassmethod

class **Test**(ABC):

a=10

*@abstractclassmethod*

def **abstractclass1**(*self*):

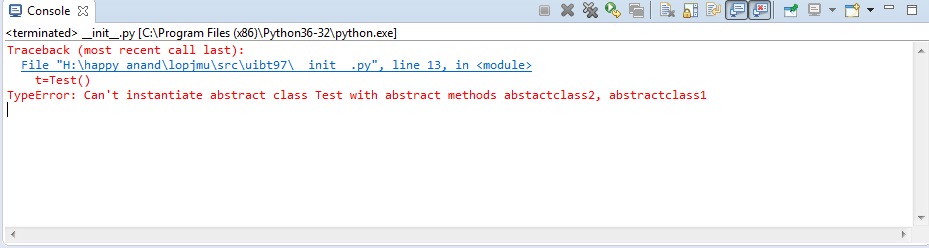
pass

*@abstractclassmethod*

def **abstactclass2**(*self*):

pass

t=Test()



2. Write a program to make the class as abstract class using ABCMeta class and create object for the class.

from abc import ABCMeta,abstractclassmethod

class **Test**(metaclass=ABCMeta):

*@abstractclassmethod*

def **abstractclass1**(*self*):

pass

*@abstractclassmethod*

def **abstactclass2**(*self*):

pass

class **T2**(Test):

def **display**(*self*):

print(*"Abstract class"*)

return

def **abstractclass1**(*self*):

print(*"Abstract class method1"*)

return

def **abstactclass2**(*self*):

print(*"Abstract class method2"*)

return

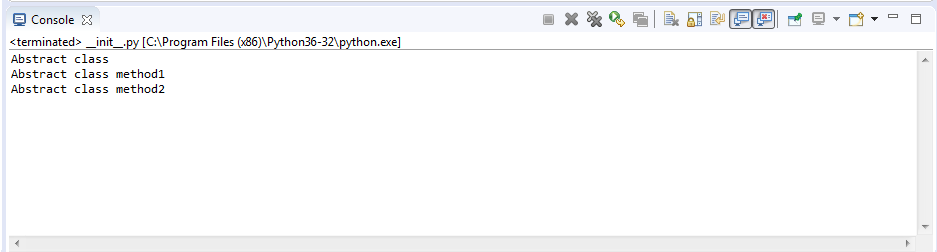
pass

t=T2()

t.display()

t.abstractclass1()

t.abstactclass2()



3. Write a program with ABC abstract class and declare some abstract method. Inherit the abstract class to another class. With out implementing the abstract method create object for child class.

from abc import ABC,abstractclassmethod

class **Test**(ABC):

*@abstractclassmethod*

def **abstractclass1**(*self*):

pass

*@abstractclassmethod*

def **abstactclass2**(*self*):

pass

class **T2**(Test):

def **display**(*self*):

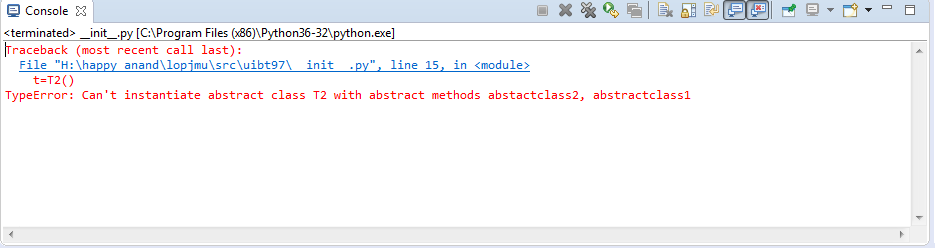
print(*"Abstract class"*)

return

pass

t=T2()

t.display()



4. Write a program with ABC abstract class and declare some abstract method. Inherit the abstract class to another class. After implementing the abstract method create object for child class and call the abstract method and normal method.

from abc import ABC,abstractclassmethod

class **Test**(ABC):

*@abstractclassmethod*

def **abstractclass1**(*self*):

pass

*@abstractclassmethod*

def **abstactclass2**(*self*):

pass

class **T2**(Test):

def **display**(*self*):

print(*"Abstract class"*)

return

def **abstractclass1**(*self*):

print(*"Abstract class method1"*)

return

def **abstactclass2**(*self*):

print(*"Abstract class method2"*)

return

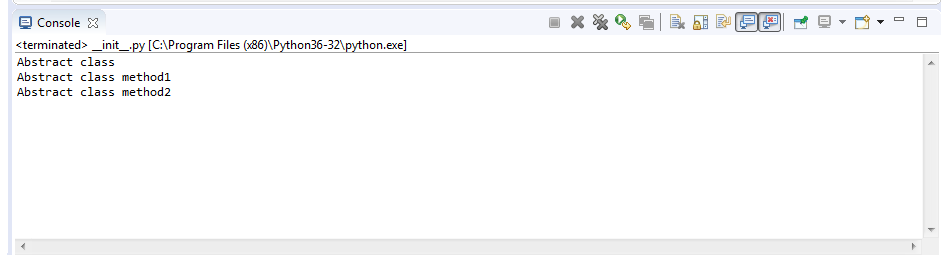
pass

t=T2()

t.display()

t.abstractclass1()

t.abstactclass2()



5. Try to implement the constructor in the abstract class.

from abc import ABC,abstractclassmethod

class **Test**(ABC):

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

print(*"Constracter"*)

*@abstractclassmethod*

def **abstractclass1**(*self*):

pass

*@abstractclassmethod*

def **abstactclass2**(*self*):

pass

class **T2**(Test):

def **display**(*self*):

print(*"Abstract class"*)

return

def **abstractclass1**(*self*):

print(*"Abstract class method1"*)

return

def **abstactclass2**(*self*):

print(*"Abstract class method2"*)

return

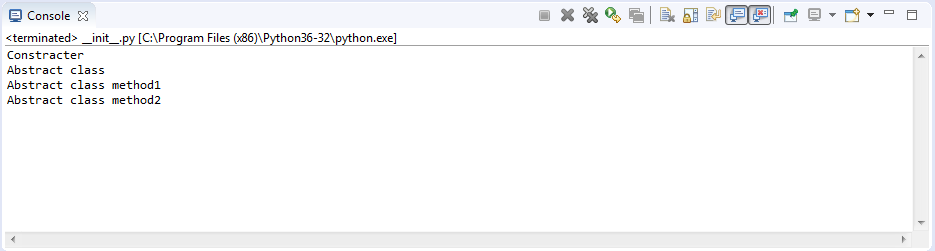
pass

t=T2()

t.display()

t.abstractclass1()

t.abstactclass2()



6. Try to implement the class variable in the abstract class.

from abc import ABC,abstractclassmethod

class **Test**(ABC):

a=10

*@abstractclassmethod*

def **abstractclass1**(*self*):

pass

*@abstractclassmethod*

def **abstactclass2**(*self*):

pass

class **T2**(Test):

def **display**(*self*):

print(*"Abstract class"*,*self*.a)

return

def **abstractclass1**(*self*):

print(*"Abstract class method1"*,*"A="*,*self*.a)

return

def **abstactclass2**(*self*):

print(*"Abstract class method2"*,*"A="*,T2.a)

return

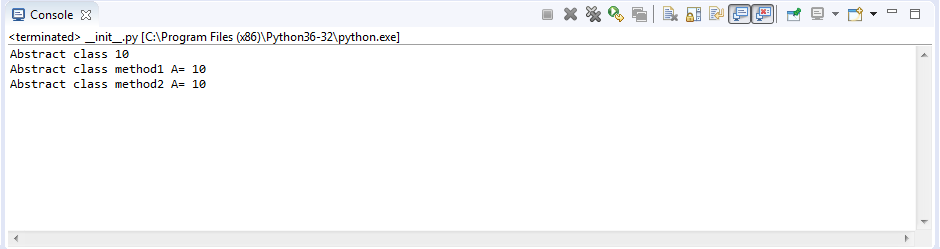
pass

t=T2()

t.display()

t.abstractclass1()

t.abstactclass2()



7. Try to implement the static method in the abstract class.

from abc import ABC,abstractclassmethod

class **Test**(ABC):

*@staticmethod*

def **staticmethod**():

print(*"Static method"*)

*@abstractclassmethod*

def **abstractclass1**(*self*):

pass

*@abstractclassmethod*

def **abstactclass2**(*self*):

pass

class **T2**(Test):

def **display**(*self*):

print(*"Abstract class"*)

return

def **abstractclass1**(*self*):

print(*"Abstract class method1"*)

return

def **abstactclass2**(*self*):

print(*"Abstract class method2"*)

return

pass

t=T2()

t.display()

t.abstractclass1()

t.abstactclass2()

t.staticmethod()

T2.staticmethod()

Test.staticmethod()

