```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


''' WAP to find the area and perimeter of a rectangle using
  classes and objects .

- Define a class to represent a bank account . Include the
  following details like name of the depositor, account number ,type of account ,
balance amount in the account . Write
  methods to assign initial values to deposit an amount ,
  Withdraw / new deposit an amount after checking the
  balance , then display name , account number , account type and current balance .

- WAP to show Hybrid Inheritance Programme in Python.

- Create an programme of Handling Exceptions With Try
   Except.

- Create an programme of Handling Multiple Exceptions
  With Multiple Excepts.

- Create an programme of Handling Exceptions With Try
  Else.

- Create an programme of User define Exceptional handling.
'''



# In[5]:


#area & perimeter of reactangleint
class rectangle:
    def __init__(self,l,b):
        self.l=l
        self.b=b
    def area(self):
        return self.l*self.b;
    def perimeter(self):
        return 2*(self.l+self.b)
r=rectangle(5,2)
print(r.area())
print(r.perimeter())


# In[39]:


#bank
class bank:
    def __init__(self):
        self.name=input("name=")
        self.acn=int(input("account no"))
        self.actype=input("account type")
        self.cb=float(input("current balance"))
```

```
class account(bank):
    def deposite(self):
        am=int(input("amount deposited"))
        self.cb=self.cb+am
        print("new balance=",self.cb)
    def withdraw(self):
        am=int(input("amount withdraw"))
        if am>self.cb:
            print("insufficient balance")
        else:
            self.cb=self.cb-am
            print("your balance remainning after withdrawling money=",self.cb)




ac=account()
ac.deposite()
ac.withdraw()



# In[ ]:




# In[9]:


#hybrid inheritance prog.
class A:
    a=0
class B:
    b=0
class C(A,B):
    c=0
    def read(self):
        self.a=int(input())
        self.b=int(input())
        self.c=int(input())
    def display(self):
        print(self.a)
        print(self.b)
        print(self.c)
c1=C()
c1.read()
c1.display()


# In[11]:


#programme of Handling Exceptions With Try Except
try:
    print("try")
```

```
    print(100/0)
    print("last line of try")

except:
    print(" except")


# In[18]:


#Create an programme of Handling Multiple Exceptions With Multiple Excepts.
try:
    num=int(input(" "))
    print(10/num)
    print("end of try")

except ZeroDivisionError:
    print("zero division")
except ArithmeticError:
    print("Arithmetic  error ")
except ValueError:
    print("value error")


# In[21]:


#Create an programme of Handling Exceptions With Try  Else.
try:
    #print("try")
    n=int(input())
    print(100/n)
    print("end try")
except:
    print("except")
else:
    print("else")


# In[27]:


#Create an programme of User define Exceptional handling.
class Marriage(Exception):

    def __init__(self,arg):
        self.name=arg
        print("your name=",self.name)
        age=int(input("Enter your age:"))

        if age<18:
         raise Marriage("To Early for marriage")

        elif age>=22 and age<=35:

         print("You are at best age for marriage")

        else:
            raise Drivingoldage("You are too old to Marriage")
```

```
c1=Marriage("s")

# In[ ]:
```