In [1]:
```python
import abc
class pwskills:

    @abc.abstractmethod
    def student_details(self):
        pass

    @abc.abstractmethod
    def student_assignment(self):
        pass

    @abc.abstractmethod
    def student_marks(self):
        pass
```

In [2]:
```python
class web_dev(pwskills):

    def student_details(self):
        return "this is a meth for taking details from students"

    def student_assignment(self):
        return "this is a meth for taking assignments from the students"
```

In [3]:
```python
class data_science_masters(pwskills):

    def student_details(self):
        return "harshit"

    def student_assignment(self):
        return "submitted"
```

In [4]:
```python
dsm = data_science_masters()
```

In [5]:
```python
dsm.student_details()
```

Out[5]:
```
'harshit'
```

In [6]:
```python
WEB_D = web_dev()
```

In [7]:
```
WEB_D.student_assignment()
```

Out[7]:
```
'this is a meth for taking assignments from the students'
```

In [ ]:
```

```

In [8]:
```
#POLYMORPHISM ke ander hamne 2 class ready ki thi, uske baad unke objects ka ek list banaya
#main :- polymorphism ke ander ek function ka alag alag input keliye alag alag behaviour rehta hai
```

In [9]:
```
#Abstraction ke ander hamne ek skeleton ready kiya jise hame baki alag alag department ke logo ko dediya or kaha ki d
# skeleton same hai lekin alag alag dept. usme apne according data input de rahe hain, baad hame jis department ka da
```

In [ ]:
```

```

In [10]:
```
class pwskills1:

    def student_details(self):
        pass

    def student_assignment(self):
        pass

    def student_marks(self):
        pass
```

In [11]:
```
class test(pwskills1):

    def student_details(self):
        return"harshit"

    def student_marks(self):
        return "90"
```

In [12]:
```
class test1(pwskills1):

    def student_details(self):
        return "krishna"
```

```
    def student_marks(self):
        return "infinity"
```

In [13]:
```
test1 = test1()
```

In [14]:
```
test = test()
```

In [15]:
```
test1.student_assignment()
```

In [17]:
```
test.student_details()
```

Out[17]:
```
'harshit'
```

In [ ]:

In [ ]:

In [ ]:

In [29]:
```
from abc import ABC, abstractmethod

class AbstractClass(ABC):
    @abstractmethod
    def abstract_method(self):
        pass

class ConcreteClass(AbstractClass):
    def abstract_method(self):
        # Provide implementation for the abstract method
        print("Implementation of abstract_method in ConcreteClass")

# Attempting to instantiate the abstract base class will raise an error
# abstract = AbstractClass()  # Raises TypeError

# Instantiate a concrete subclass
concrete = ConcreteClass()
concrete.abstract_method()  # Outputs: "Implementation of abstract_method in ConcreteClass"
```

```
Implementation of abstract_method in ConcreteClass
```

In [ ]:

In [25]:
```
class abstractclass:

    def abstract_method(self):
        pass
```

In [26]:
```
class concreteclass(abstractclass):

    def abstract_method(self):
        return "implementation of abstract method in concrete class"
```

In [27]:
```
concrete = concreteclass()
```

In [28]:
```
concrete.abstract_method()
```

Out[28]:
```
'implementation of abstract method in concrete class'
```

In [ ]:

In [ ]: