# Python - Abstração

A abstração é um dos princípios importantes da programação orientada a objetos. Refere-se a uma abordagem de programação pela qual apenas os dados relevantes sobre um objeto são expostos, ocultando todos os outros detalhes. Essa abordagem ajuda a reduzir a complexidade e aumentar a eficiência no desenvolvimento de aplicações.

Existem dois tipos de abstração. Uma delas é a abstração de dados, em que a entidade de dados original é ocultada por meio de uma estrutura de dados que pode funcionar internamente por meio das entidades de dados ocultas. Outro tipo é chamado de abstração de processo. Refere-se a ocultar os detalhes de implementação subjacentes de um processo.

Na terminologia de programação orientada a objetos, uma classe é considerada uma classe abstrata se não puder ser instanciada, ou seja, você pode ter um objeto de uma classe abstrata. No entanto, você pode usá-lo como classe base ou pai para construir outras classes.

Para formar uma classe abstrata em Python, ela deve herdar a classe ABC que está definida no módulo abc. Este módulo está disponível na biblioteca padrão do Python. Além disso, a classe deve ter pelo menos um método abstrato. Novamente, um método abstrato é aquele que não pode ser chamado, mas pode ser substituído. Você precisa decorá-lo com o decorador @abstractmethod.

## Exemplo

```
from abc import ABC, abstractmethod
class demo(ABC):
   @abstractmethod
   def method1(self):
      print ("abstract method")
      return
   def method2(self):
      print ("concrete method")
```

A classe **demo** herda a classe ABC. Existe um método1() que é um método abstrato. Observe que a classe pode ter outros métodos não abstratos (concretos).

Se você tentar declarar um objeto da classe demo, Python gera TypeError -

```
  obj = demo()
      ^^^^^^
TypeError: Can't instantiate abstract class demo with abstract method method1
```

A classe **demo** aqui pode ser usada como pai para outra classe. No entanto, a classe filha deve substituir o método abstrato da classe pai. Caso contrário, Python gera este erro -

```
TypeError: Can't instantiate abstract class concreteclass with abstract method method1
```

Portanto, a classe filha com o método abstrato substituído é fornecida no **exemplo** a seguir -

```
from abc import ABC, abstractmethod
class democlass(ABC):
   @abstractmethod
   def method1(self):
      print ("abstract method")
      return
   def method2(self):
      print ("concrete method")

class concreteclass(democlass):
   def method1(self):
      super().method1()
      return

obj = concreteclass()
obj.method1()
obj.method2()
```

## Saída

Quando você executa este código, ele produzirá a seguinte saída -

```
abstract method
concrete method
```