Background

- ✓ Both data hiding and data encapsulation are essential concepts of object-oriented programming.
- ✓ **Data encapsulation** hides the private methods and class data parts, whereas **Data hiding** only hides class data components.
- ✓ **Encapsulation** wraps up the complex data to present a simpler view to the user, whereas **Data hiding** restricts the data use to assure data security.
- ✓ Data hiding also helps to reduce the system complexity to increase the robustness by limiting the interdependencies between software components.
- ✓ In python, **there exists public, private and protected access specifiers** (used in inheritance). **Data hiding is achieved by using the private access specifier.**

Data Hiding

- ➢ Data hiding is the technique of **declaring the data within a class as private to prevent its direct access from outside the class.**
- ➢ Data hiding is the process of **making the data private so that it can be accessed only by the class where it is declared.**
- ➢ **It guarantees to maintain data integrity** by restricting the data access to class members.
- ➢ **It imparts security, along with discarding dependency** as the user remains isolated from the program implementation part.
- ➢ Data hiding is **also known as information hiding**.

How to hide the data (variable)?

To hide the variable, we need to write double underscore (_ _) before the variable name.

Consider the following code:

```
class MyClass:
    num = 10
    def add(self, a):
        sum = self.num + self.a
        print(sum)

ob = MyClass()
ob.add(20)
ob.num
```

Now, to make the variable private, it is preceded by double underscore as:

Example:

```
class MyClass:
    __num = 10
    def add(self, a):
        sum = self.__num + a
        print(sum)

ob = MyClass()
ob.add(20)
ob.__num        #This gives an error, the object is trying to access the private variable outside class.
```

Output:

```
30
Error
```

To access hidden data?

To rectify the error, we can access the private member through the object_name single underscore and the class_name and the hidden data:

Example:

```
class MyClass:
      __num = 10
      def add(self, a):
           sum = self.__num + a
           print(sum)

ob = MyClass()
ob.add(20)
ob._MyClass__num
```

Output:

30

How to hide the method (function)?

To hide any function, the function name is preceded by two double underscore sign while declaration.

Example:

```
class SeeMee:
  def youcanseeme(self):
    print('you can see me')

  def __youcannotseeme(self):
    print('you cannot see me')

#Outside class
Check = SeeMee()
Check.youcanseeme()
Check._SeeMee__youcannotseeme()
```

Output:

you can see me

you cannot see me

Example of Data Hiding:

Example 1:

```
class Person:
  def __init__(self):
    self.name = 'Manjula'
    self.__lastname = 'Dube'

  def PrintName(self):
    print(self.name , end= " ")

#Outside class
P = Person()
P.PrintName()
P._Person__lastname
```

Output:

Manjula Dube

Example 2:

```
class Solution:
    __privateCounter = 0

    def sum(self):
        self.__privateCounter += 1
        print(self.__privateCounter)
count = Solution()
count.sum()
count.sum()
count.__privateCount

# Here it will show error because it unable to access private member
```

Output:

```
1
2
Traceback (most recent call last):
  File "<string>", line 10, in <module>
AttributeError: 'Solution' object has no attribute '__privateCount'
```

To rectify the error:

```
class Solution:
    __privateCounter = 0

    def sum(self):
        self.__privateCounter += 1
        print(self.__privateCounter)
count = Solution()
```

```
count.sum()
count.sum()
count._Solution__privateCounter
```

Output:

```
1
2
2
```

Example 3:

```
class MyClass:
    __hiddenVariable = 0

    def add(self, increment):
        self.__hiddenVariable += increment
        print(self.__hiddenVariable)

myObject = MyClass()
myObject.add(2)
myObject.add(5)
myObject.__hiddenVariable
```

Output:

```
2
2
7
Traceback (most recent call last):

File "<string>", line 18, in <module>

AttributeError: 'MyClass' object has no attribute '__hiddenVariable'
```

To remove error,

```
class MyClass:

    __hiddenVariable = 10


myObject = MyClass()
myObject._MyClass__hiddenVariable
```

Output:

```
10
```

Advantages of Data Hiding:

1. It helps to prevent damage or misuse of volatile data by hiding it from the public.
2. The class objects are disconnected from the irrelevant data.
3. It isolates objects as the basic concept of OOP.
4. It increases the security against hackers that are unable to access important data.

Disadvantages of Data Hiding:

1. It enables programmers to write lengthy code
2. It hides important data from within a class itself.
3. The linkage between the visible and invisible data makes the objects work faster, but data hiding prevents this linkage.

Example: We can understand data hiding with an example. Suppose we declared an **Account** class with a data member **balance** inside it. Here, the account balance is sensitive information. We may allow someone to check the account balance but won't allow altering the balance attribute. So, by declaring the balance attribute private, we can restrict the access to balance from an outside application.