British Journal of Computer, Networking and Information Technology
ISSN: 2689-5315
Volume 6, Issue 1, 2023 (pp. 83-93)
www.abjournals.org

DETAILED STUDY OF THE OBJECT-ORIENTED PROGRAMMING (OOP) FEATURES IN PYTHON

Nzerue-Kenneth Peace Ezinne, Onu Fergus Uchenna, Denis Ashishie Undiukeye,

Igwe Joseph Sunday and Ogbu Nwani Henry

Department Of Computer Science, Ebonyi State University, Abakaliki - Nigeria

Cite this article:

Nzerue-Kenneth P.E., Onu F.U., Denis A.U., Igwe J.S., Ogbu N.H. (2023), Detailed Study of the Object-Oriented Programming (OOP) Features in Python. British Journal of Computer, Networking and Information Technology 6(1), 83-93. DOI: 10.52589/BJCNIT-FACSOJAO

Manuscript History

Received: 18 Sept 2023

Accepted: 7 Nov 2023

Published: 29 Nov 2023

ABSTRACT: *Human beings are naturally classified; OOP is closely aligned to how human brains work. The Mathematical functional approach is a more rigorous way to capture an idea; it is more difficult to transpose and the code is not easily readable. To overcome the limitations of procedural, structural, and functional programming, OOP was developed. Because of its versatility, it supports various programming styles which include maintenance, addition and overriding; however, modification of existing code is made easier. OOP programs prevent you from repeating code, making developers choose OOP in their python program because a class can be defined once and reused many times (Thelin, 2020). By the end of this article, you will be able to create classes, instantiate objects from them, and integrate the four methods of OOP by creating modules of Python projects. With the basic concept of OOP in Python, this article has shown how we can hide our data by making it private (abstraction), allowing for code reusability (inheritance), constraining dependency management through polymorphism and rapping data and function together to prevent data from being accessed by the code outside this shield (encapsulation).*

KEYWORDS: OOP Object Oriented Programming, features of OOP, Inheritance, Encapsulation, Abstraction, Polymorphism.

Article DOI: 10.52589/ BJCNIT-FACSOJAO
DOI URL: https://doi.org/10.52589/ BJCNIT-FACSOJAO

British Journal of Computer, Networking and Information Technology
ISSN: 2689-5315
Volume 6, Issue 1, 2023 (pp. 83-93)

www.abjournals.org

INTRODUCTION

In the 1930s, lambda calculus, and the Turing machine explored the first functional and imperative programming paradigms mathematically. In the 1940s, low-level language like assembly and machine code appeared. Towards the end of the 1950s, high-level languages and their imperatives, such as FORTRAN and COBOL, appeared. Both functional programming and imperative programming are rooted in the mathematics of computation theory. To help to manage complexity in large programs and solve the rigorous way to capture an idea in functional and imperative programming paradigms, Alan Kay circa in 1966 coined "Object-Oriented Programming" (OOP). Object-oriented programming is the use of predefined programming modular units (objects, classes, subclasses, and so forth) in order to make programming faster and easier to maintain (Hemmendinger, 2008).

Object Oriented Programming is a programming paradigm that uses the concept of classes and objects to create models based on the real-world environment which helps the programmer to develop complex software (Ryan, 2020). Here, software programs are simplified into reusable pieces of code called classes. These classes are used to create individual instances of an object. Examples of OOP include the following: Python, JavaScript, C++, Java, and many more.

Object-oriented programming helps us represent things in our code and OOP is a methodology or style of managing your code. It was invented to better organize your code in logical units which are called classes and objects. Before the invention of OOP, procedural code and spaghetti code were in use. Functional programming thinks of programming as Maths while OOP thinks of programming as modeling; the code is more navigable because it is closer to the problem. It allows you to explore problems in small chunks in a way that is closely related to how we think about the problem. The OOP programming paradigm was developed in order to overcome the limitations of procedural programming (Learn Computer Science, 2021).

One of the challenges of programming is how to represent data in our code. To represent straightforward data in our code is very easy, for instance, storing integers, floats, or even a list. But what if you want to represent something a little bit more complex? That is where object-oriented programming comes into play in Python programming. More so, python offers various tools for performing cluster analysis of data. In clustered data, observations within a cluster show similarity between themselves because they share common features different from observations in the other clusters. In a given population, different clustering may surface because correlation may occur across more than one dimension (Yahya *et al.*, 2023). There are three techniques for how to form clusters in python. These techniques include spectral clustering, K-means clustering and Gaussian mixture models.

The core purpose of this paper is to elaborate on the impact of OOP in Python. This is achieved by exploring the four concepts of OOP, which include the following: Encapsulation, Inheritance, Polymorphism, and Abstraction.

Article DOI: 10.52589/ BJCNIT-FACSOJAO
DOI URL: https://doi.org/10.52589/ BJCNIT-FACSOJAO

British Journal of Computer, Networking and Information Technology
ISSN: 2689-5315
Volume 6, Issue 1, 2023 (pp. 83-93)
www.abjournals.org

Classes and Object

Classes: A class is a blueprint or template through which objects are being created. The class consists of attributes (properties), functions (methods), and behavior. Everything in Python is an object. All the objects have a type; these types are declared using Classes. Properties are defined in a class and it is possible to assign values to properties while the object is being instantiated. The properties are collectively known as the state of the objects creating a simple class.

Example of a class in Python:

```
class Parent:
    def __init__(self, fname, lname, age):#Create a parent cl
        self.first_name = fname
        self.second_name = lname
        self.age = age
```

To create a class in Python, first, we use a "Class" keyword to begin and set its properties.

In the example above, we have created a class named Parent. It has three data attributes which are fname, name, and age. We have also defined a method called function below. We can create an instance of the person class object just as seen below:

Objects: Objects are the real thing; they are created from the class name. Objects can contain methods and these methods are functions that belong to the objects. Creating an object in a class is called instantiation; properties can be assigned values, making each object a type of unique entity. For example, we use Parent to create an object like this:

```
class Parent:
    def __init__(self, fname, lname, age):
        self.first_name = fname
        self.second_name = lname
        self.age = age

    def function (self):# Parent method
        print(self.first_name, "Is coding")
        print("she's " + str (self.age) + "years old")
p1 = Parent("Peace", "Dannis", 30) #create parents object
p1.function()
```

```
Parent › function()
Run: object
Peace Is coding
she's 30years old
```


Article DOI: 10.52589/ BJCNIT-FACSOJAO
DOI URL: https://doi.org/10.52589/ BJCNIT-FACSOJAO

British Journal of Computer, Networking and Information Technology
ISSN: 2689-5315
Volume 6, Issue 1, 2023 (pp. 83-93)

www.abjournals.org

1. Encapsulation
2. Inheritance
3. Polymorphism
4. Abstraction.

Abstraction:

This means showing the necessary details to the user of a computer. An abstract class can be considered as a blueprint for other classes (GeeksforGeeks, 2021). In abstraction, we care about calling the method not the interpretation, just like remote TV; you are just pressing the button to change the channel but not taking care of the internal information like what will happen when you change the channel. Abstraction means to simplify reality. Abstraction means data hiding. It is the act of representing the essential feature without knowing the background details. Here, we can show necessary information and hide the unnecessary part, e.g., in a car, the user is interested in starting the car without knowing what is happening inside the engine of the car or what happens when you press the brake of a car because that is not necessary for you.

Inheritance

Inheritance is a powerful future that allows you to code reusability. Inheritance is the ability of a class to inherit methods and/or attributes of another class (Lakshmanamoorthy, 2021). This allows you to extend the class instead of typing the code again and again. Inheritance allows us to create a class from another class; it minimizes the working time of a programmer and reduces errors and repetition. In inheritance, the object of a class acquires some properties of another class.

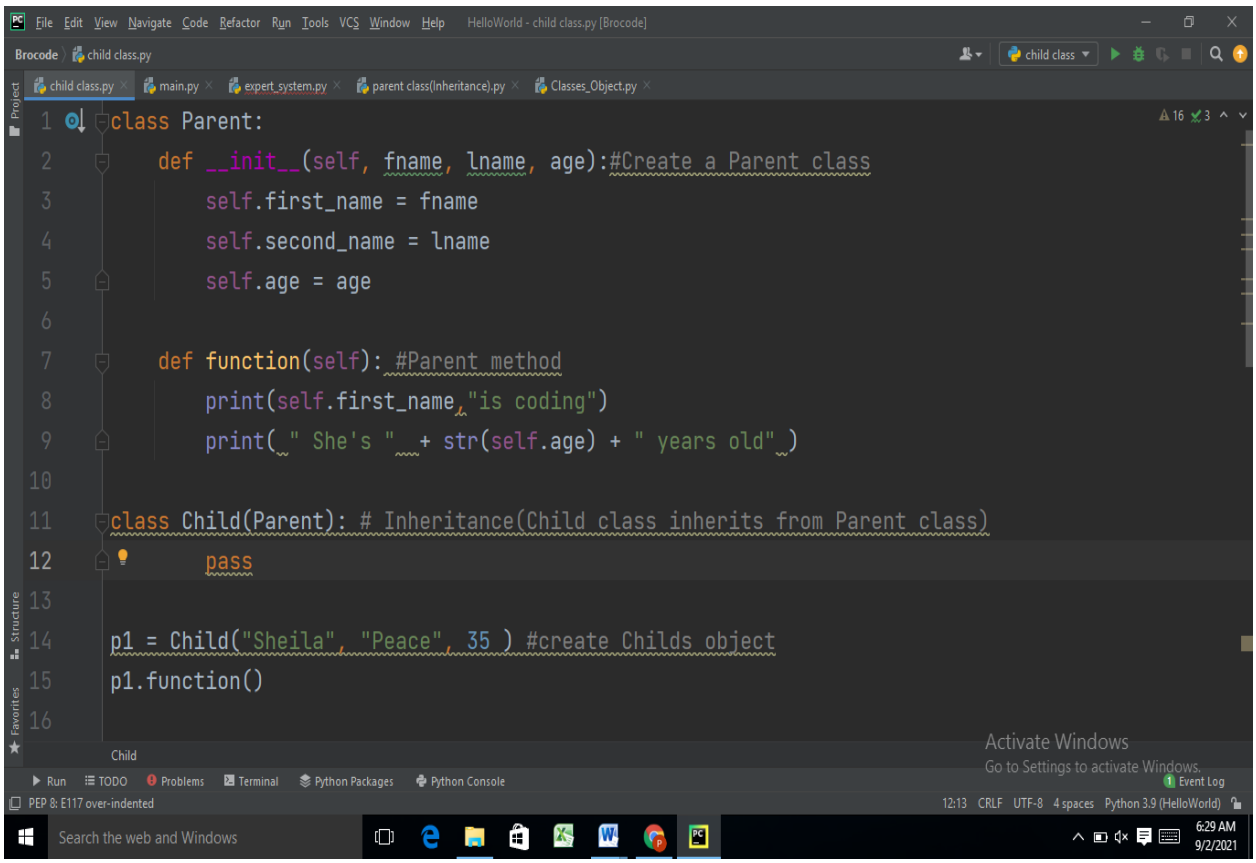
- The class that is inherited from is known as the base class or super class
- The class that is inherited to is known as a derived class or subclass.

For example, we can create a child's class from the parent class or the super class; as new objects are created, they inherit characteristics from existing ones and even add their own features also, just as seen below:

Article DOI: 10.52589/ BJCNIT-FACSOJAO
DOI URL: https://doi.org/10.52589/ BJCNIT-FACSOJAO

British Journal of Computer, Networking and Information Technology
ISSN: 2689-5315
Volume 6, Issue 1, 2023 (pp. 83-93)
www.abjournals.org

Impact of Inheritance and Its application in Python



In this example, the child's class copies the whole attribute of the parent's class: both data and methods from the parent's class. This concept is called inheritance. Composition is generally favored over class inheritance because of the following enhanced flexibility, reduced complexity, clearer interfaces, and promote code reuse (Elliott, 2023).

Note: A child class can add more new attributes outside the ones inherited from the parent's class.

Types of Inheritance in Python

The derived class may inherit all properties or some of the properties from the super class. Equally, a class can inherit properties from more than one class or more than one level generation. Typically, inheritance is classified into five categories (Rout, 2020):

- Single inheritance
- Multiple inheritance
- Multilevel inheritance

Hierarchical inheritance

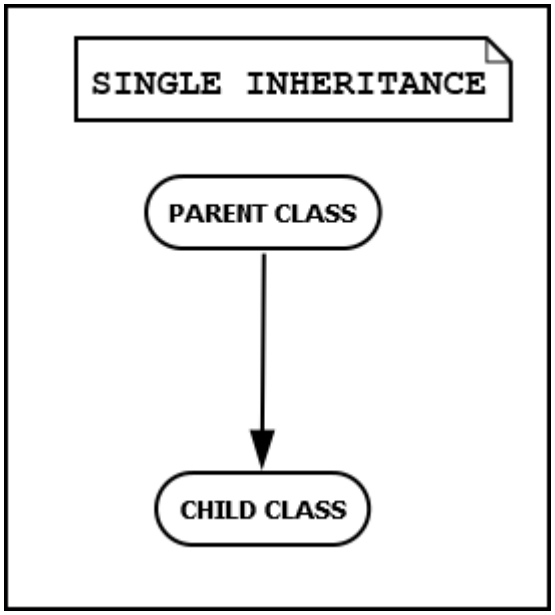
- Hybrid inheritance.

Article DOI: 10.52589/ BJCNIT-FACSOJAO
DOI URL: https://doi.org/10.52589/ BJCNIT-FACSOJAO

British Journal of Computer, Networking and Information Technology
ISSN: 2689-5315
Volume 6, Issue 1, 2023 (pp. 83-93)
www.abjournals.org

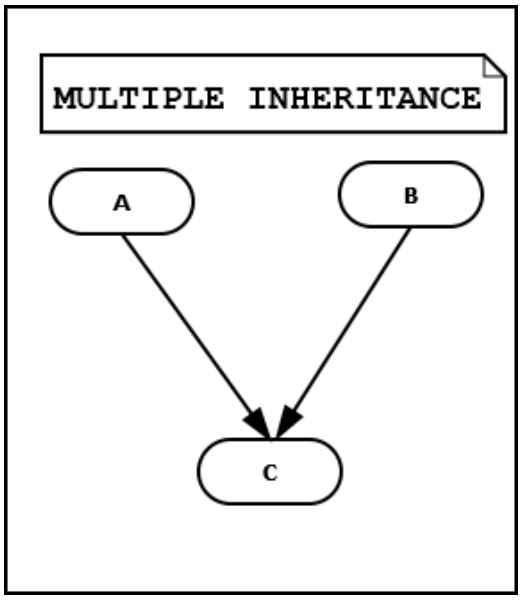
Single Inheritance

Single inheritance enables a derived class to inherit properties from a single parent class, thus enabling code reusability and the addition of new features to existing code (kumar_satyam 2023). Example is shown below:



Multiple Inheritance

When a class is derived from more than one base class, this type of inheritance is called multiple inheritances. All the properties of the base classes are carried over to the derived class. Below is an illustration of a multiple inheritance class:

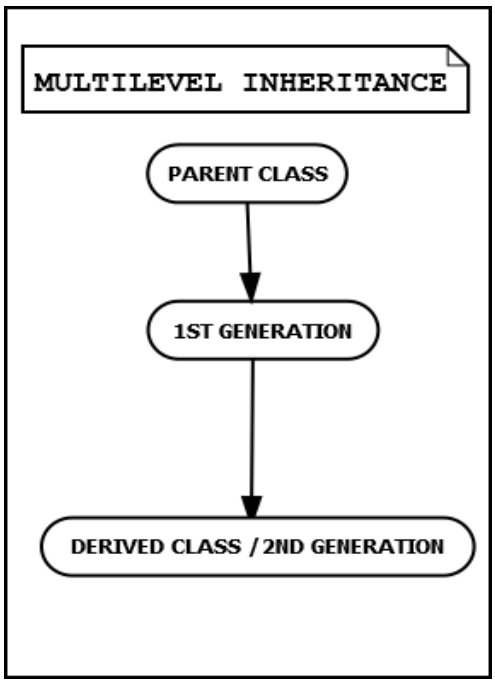


Article DOI: 10.52589/ BJCNIT-FACSOJAO
DOI URL: https://doi.org/10.52589/ BJCNIT-FACSOJAO

British Journal of Computer, Networking and Information Technology
ISSN: 2689-5315
Volume 6, Issue 1, 2023 (pp. 83-93)
www.abjournals.org

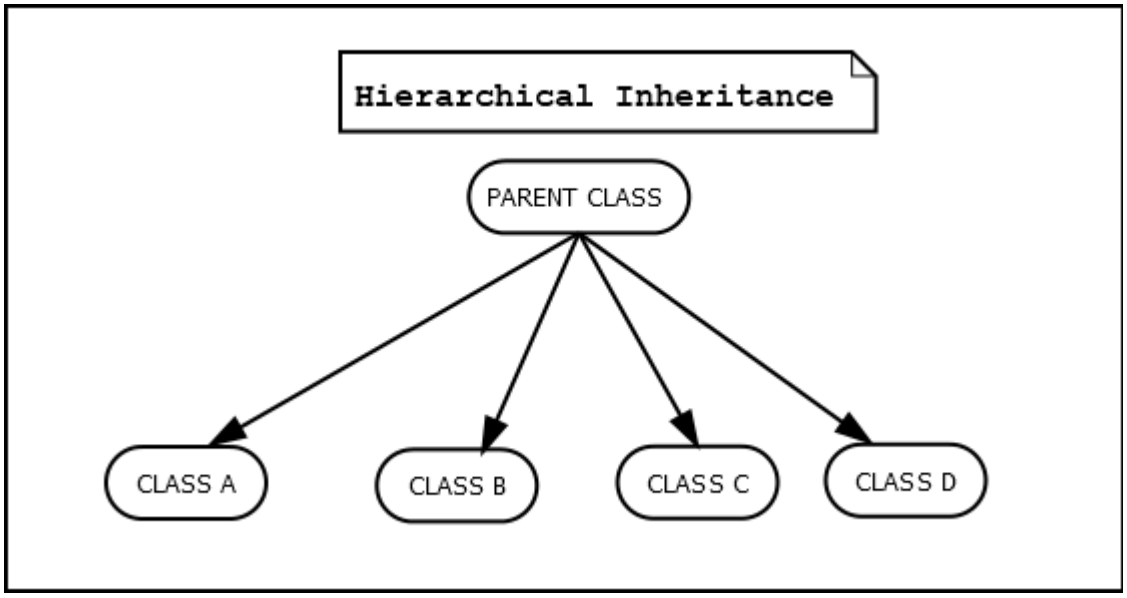
Multilevel Inheritance

In multilevel inheritance, the properties of the derived class and the based classes are inherited further into the new derived class. In other words this is referred to as the 2nd generation class.



Hierarchical Inheritance

Hierarchical inheritance occurs when multiple derived class are created from a single base class, as shown below:



Article DOI: 10.52589/ BJCNIT-FACSOJAO
DOI URL: https://doi.org/10.52589/ BJCNIT-FACSOJAO

British Journal of Computer, Networking and Information Technology
ISSN: 2689-5315
Volume 6, Issue 1, 2023 (pp. 83-93)
www.abjournals.org

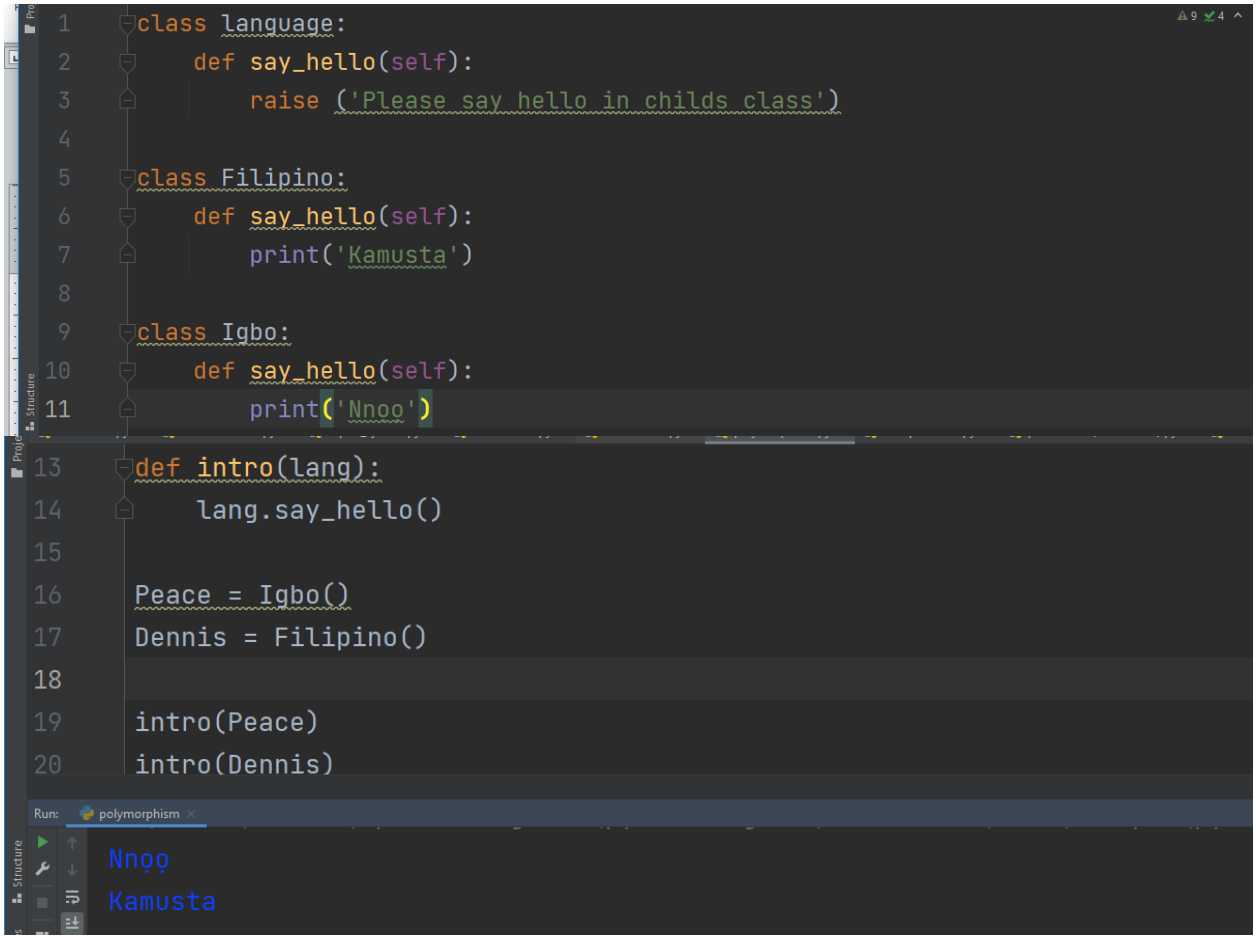
Hybrid Inheritance

This consists of various types of inheritance:

Polymorphism

This comes from the word poly which means many and morph which means form. Polymorphism is the ability of an object to take many forms. OOP constraints dependency management through polymorphism. An operation may show entirely different behaviors for different data sets. Basically, it allows you to determine what kind of function to run while the program is running. Objects are designed to share behaviors and they can take on more than one form (Gillis & Lewis, 2021). Polymorphism means changing from one form to another form. For example, a man is a husband, a father, and at the same time a son of a woman. In this example, the man plays a different role in different situations. To explain it further, let us look at this example—"Hello" can be spoken in different languages just as seen below:

Impact of Polymorphism and Its Application in Python



Article DOI: 10.52589/ BJCNIT-FACSOJAO
DOI URL: https://doi.org/10.52589/ BJCNIT-FACSOJAO

British Journal of Computer, Networking and Information Technology
ISSN: 2689-5315
Volume 6, Issue 1, 2023 (pp. 83-93)

www.abjournals.org

Encapsulation

This is a system of rapping data and functions together. In other words, to hide the complexity of objects from the programs and those using them, the implementation of those codes should be hidden and this simplifies the use of the things. Encapsulation was built on the idea of data hiding or information hiding. This is where we restrict access to certain properties and methods of an object to whatever is calling that object.

Encapsulation refers to the rapping up of data under a single unit. It is a mechanism that binds code and the data it manipulates. Another way to think about encapsulation is that it is a protective shield that prevents data from being accessed by the code outside this shield and in this, the variable or the data of a class is hidden from any other class and can be accessed only through any member function of the own class in which they are declared. For example, in a capsule that envelops a drug; similarly, in encapsulation, the methods and the variables of a class are very well hidden and safe. Another example would be a vending machine die.

Benefits of Encapsulation

- **Data Hiding:** Here, the user may have no idea about the inner implementation of the class and even the user will not be aware of how the class is storing the value in the variables. They will only see what you want them to see which is basically passing the value to a method and then the variables are getting initialized with that value. So they do not really know what is going on inside the class and that is one benefit of encapsulation.

- **Increased Flexibility:** Here, we can make the variables of the class you know according to how you want them to be used, in case we just want it to be a read-only method or you know it can be write-only depending on what we require.

- **Reusability:** With reusability, it becomes easy to change with new requirements.

Major Differences Between Abstraction and Encapsulation

Encapsulation is known for wrapping the code and data together whereas abstraction basically provides the abstract, that is, it gives you the most necessary details in the data and it is only the useful details that you will be seeing in abstraction. However, in encapsulation, the whole code and all the necessary information are wrapped together. Encapsulation focuses on how the implementation is going to work. Abstraction gives you the information you need from your data.

In Python, plain attributes are used in order to achieve encapsulation. This is done with the help of naming conventions through which we are able to distinguish between protected and private members, so we add double underscores in front of the variables and any function name. It can hide them when accessing them from out of the class or scope of the class. In other words, python does not have any private methods but we can use the double underscore naming convention by which we can name them private methods and sort them off. Note, a single underscore makes a variable private but if you do not want anyone to change it, you need to use a double underscore. For you to define it and access it, you need to define getter for instance.

Article DOI: 10.52589/ BJCNIT-FACSOJAO
DOI URL: https://doi.org/10.52589/ BJCNIT-FACSOJAO

British Journal of Computer, Networking and Information Technology
ISSN: 2689-5315
Volume 6, Issue 1, 2023 (pp. 83-93)
www.abjournals.org

Impact of Encapsulation and Its Application in Python

```python
class Amount:
    def __init__(self, price):
        self.__last_price = price + price * 0.07 # note this last_price is pu
                                                 # make it private put d
    def get_last_price(self):                    # that no one can chang
        return self.__last_price

    def set_last_price(self,coupon_discount): # if your given a coupon u can
        self.__last_price = self.__last_price - self.__calculate_coupon_disco

    def __calculate_coupon_discount(self,coupon_discount):
        return self.__last_price *(coupon_discount/100)
```

Parameter price of Encapsulation.Amount.__init__
price: {__add__, __mul__}
HelloWorld

```python
cloth = Amount(30)
cloth.set_last_price(30)
print(cloth.get_last_price())
```

```
Encapsulation
C:\Users\IPPIS1\PycharmProjects\pythonProject\HelloWorld\venv\Scripts\python.ex
22.47
```

CONCLUSION

To design quality, easy-to-maintain, and robust software in Python, these four pillars are the key ingredients. OOP in Python helps to code reusability, simplification (simple to manage), easy to maintain, and clear structure.

With object-oriented programming, it is very easy to work with objects inside Python, declared classes, and store your data

To overcome the limitations of procedural programming, OOP was developed. Maintenance and modification of existing code are made easier in Python because of the impact of OOP as new objects are created and they inherit characteristics from existing ones.

OOP is more secure because it allows the developer to protect the key data elements in Python. This can be done by restricting data access to only methods that belong to that particular object; this is called abstraction.

More so, OOP is less complex because of its modularity in its program, thereby making it easy to create new data objects from existing objects. This concept makes object-oriented programming easy to modify. To develop comprehensive and well-designed software products in Python, it is paramount to have a clear understanding of OOP.

Article DOI: 10.52589/ BJCNIT-FACSOJAO
DOI URL: https://doi.org/10.52589/ BJCNIT-FACSOJAO

REFERENCES

[1] R. Lakshmanamoorthy,"Object-Oriented Programming with Python", Analytics India Magazine. (2021).
[2] E. Elliott,"The Future of Programming: AI and Interface-Oriented Languages", written by Javascript Scene (2023)
[3] LCSO,"Object Oriented Programming | OOP principles", learncomputerscienceonline (2021)
[4] A.S. Gillis & S.Lewis, "What is object-oriented programming", Technical Writer and Editor, techtarget 2021
[5] D. Team,"Python Project for Beginners – Alarm Clock. DataFlair", Blog training (2021)
[6] D.Hemmendinger,"Object-oriented programming | Object-oriented programming | Classes, Encapsulation, Polymorphism", The Editors of Encyclopedia Britannica(Britannica) (2008)
[7] Ryan Thelin,"How to Use Object-Oriented Programming in Python", Educative Blog. issued September 6, 2020)
[8] GeeksforGeeks, " Abstract classes in Python", GeeksforGeeks (2021).
[9] R. S. Rau, "Object-oriented programming (OOP)". International Research Journal of Engineering and Technology (IRJET) (2020)
[10] Kumar_satyam," Python inheritance-Python OOP-Python OOP Concept", geeksforgeeks retrieved(12 August 2023)
[11] W.B. Yahya, Y Bello, & A Abdulraheem,"Model Fitness and Predictive Accuracy in Linear Mixed-Effects Models with Latent Clusters", Journal of the Nigerian Society of PhysicalSciences 5 (2023) 1392.

Article DOI: 10.52589/ BJCNIT-FACSOJAO
DOI URL: https://doi.org/10.52589/ BJCNIT-FACSOJAO