BRIGHT MIND
PUBLISHING

Educator Insights: A Journal of Teaching Theory and Practice
Volume 01, Issue 05, May, 2025
brightmindpublishing.com
ISSN (E): 3061-6964

FUNDAMENTAL PROPERTIES AND EXAMPLES OF CLASS USAGE IN THE PYTHON PROGRAMMING LANGUAGE

Saidaxmedov Eldor Islomovich
Denov tadbirkorlik va pedagogika instituti
“Axborot Texnologiyalari“ kafedrasining o‘qtuvchisi
Email: e.saidaxmedov@dtpi.uz
UDK 371.3:004:373.3
ORCID 0009-0001-4349-1765

Boboyorov Novruzbek Inotullayevich
Denov tadbirkorlik va pedagogika instituti
“Axborot Texnologiyalari“ kafedrasi magistri
Email:inotullayevnavruz@gmail.com

Nabiyev Raxim Azamatovich
Denov tadbirkorlik va pedagogika instituti
“Axborot Texnologiyalari“ kafedrasi o’qituvchisi
Email:raximnabiyev9@gmail.com

Abstract
Classes in the Python programming language are a central concept of object-oriented programming. This article explores the fundamental characteristics of classes, including attributes for storing data and methods for defining behavior. It also provides practical examples of creating classes, instantiating them (creating objects), and using them to solve various programming problems. Important principles of object-oriented programming such as inheritance, polymorphism, and encapsulation are also analyzed in detail. Throughout the article, the significance of classes in the software design and development process is demonstrated through real-life scenarios

Keywords: Python Classes, Object-Oriented Programming (OYD), Attributes, Methods, Objects, Instantiation, Inheritance, Polymorphism, Encapsulation, Abstraction, Software Design.

BRIGHT MIND PUBLISHING

Educator Insights: A Journal of Teaching Theory and Practice
Volume 01, Issue 05, May, 2025
brightmindpublishing.com
ISSN (E): 3061-6964

Introduction

Python programming language is one of the most popular and powerful tools in modern programming and is widely used in a variety of industries. Among its many advantages, its full support of the principles of Object Oriented Programming (OYD) stands out. Classrooms, on the other hand, provide invaluable opportunities for organizing software, reusing code, and modeling complex systems. This article in-depth organizes the basic properties of classes in the Python programming language, their attributes and methods. It also demonstrates through vivid examples the creation of classes, the instantiation of objects and their practical value in solving various software problems. In addition, important principles of OYD such as heredity, polymorphism, and encapsulation are analyzed, revealing the role of classes in the software design and development process [1].

LITERATURE REVIEW

The Python programming language is one of the most dynamic and widely used languages in modern programming and offers full support for the principles of Object Oriented Programming (OYD). The concept of a class is a central element of Python OYD, enabling software modulation, code reuse, and the efficient design of complex systems. This literature review examines the fundamental features of Python classes, their attributes and techniques, the process of object creation, as well as the basic principles of OYD such as inheritance, polymorphism, and encapsulation. In addition, the main approaches in the domestic and foreign literature on the application of classes in various programming fields are analyzed. Python classes are structures that combine data (attributes) and behaviors (methods) into a cohesive whole. Whereas attributes are variables that represent the state of an object, methods are functions that an object can perform. Creating a class starts with a class keyword, followed by the name of the class and two dots (:). Within the classroom, attributes and methods are defined. An object is a copy of a class and has the attributes and techniques defined in the class. The literature created by specialists from Uzbekistan is an important resource for studying and applying classes in the Python programming language. The book "Fundamentals of Python Programming" by professor D.Q. Kurbanov gives fundamental concepts of the Python language, as well as basic knowledge of classes and objects. The book explains the process of creating

Educator Insights: A Journal of Teaching Theory and Practice
Volume 01, Issue 05, May, 2025
brightmindpublishing.com
ISSN (E): 3061-6964
Licensed under* CC BY 4.0 *a Creative Commons Attribution 4.0 International License.

classes, defining attributes and methods, and instantiating objects[6]. The textbook "Object-oriented programming in Python" by associate professor F.T. Soliev examines in detail the basic principles of OYD, including encapsulation, inheritance and polymorphisms in Python [4]. The author explains each principle through practical examples, helping students to absorb these concepts in depth. And the book "Python Programming Language: Practical Exercises" by senior lecturer N.R. Mahmudov gives many practical examples of using classrooms to solve various software problems [7]. The book demonstrates the application of classes, particularly in areas such as data structures, graphical interfaces, and web programming. This local literature is important for students and programmers learning the Python programming language to strengthen the theoretical foundations of the classroom and to develop the skills to apply them in practice.
There is a lot of authoritative foreign literature on the topic of classes in the Python programming language. The book Learning Python by Marc Lutz is one of the most recommended resources for learning Python, and it explains the concept of classes and objects in a detailed, easy-to-understand way[5]. The author also discusses advanced topics such as different ways of creating classes, succession mechanisms, and operator reloading. Luciano Ramalho's book, "Fluent Python," goes deep into the specifics of Python and provides tips on how to apply classes in a more effective and "pythonic" style. In particular, the book discusses the concepts of data models, attribute management, and object-oriented design [1]. The Python Cookbook by David Beazley and Brian K. Jones, on the other hand, provides many recipes and examples of how to solve a variety of applied programming problems using classes. The book offers, in particular, practical solutions for encapsulating data, the organization of interactions between objects and the creation of complex data structures[2]. This foreign literature provides invaluable resources for in-depth study of Python programming language classes, mastering advanced techniques and their effective application in various software projects[3].
The analyzed domestic and foreign literature confirms the fundamental importance of classes in the Python programming language. The concept of the classroom is central to the organization of software, code reuse, and the design of complex systems. The main properties of classes, methods of their creation and application, as well as the basic principles of OYD are described in detail in various literature. These resources are essential resources for every programmer

learning the Python programming language to gain in-depth knowledge of the classes and to successfully apply them in practice.

RESULTS AND DISCUSSIONS

Python is an object-oriented programming language. A class is like an object constructor or a "blueprint" for creating objects. Create a class Use the class keyword to create a class:
Create a class named MyClass, add x variable:

```
class MyClass: x = 5
```

Create an object named P1 and print the value x:

```
p1 = MyClass()
print(p1.x)
```

To understand the meaning of classes, we need to understand the init() function that is built. All classes have a function called init(), which is always done when the class starts. To assign values to object properties, use the init() function or some other operation you need to perform when the object is created:
Example: Create a class named personality, use the init() function to set values for name and age:

```
class Person:
Def init (self, name, before): self.name = name
self.age = age
p1 = Person("John", 36)
print(p1.name)
print(p1.age)
```

_ The str() function controls what should be returned when a class object is specified as a string. When the str() function is set, the object returns a string result.

```
class Person:
Def init (self, name, before): self.name = name
self.age = age
p1 = Person("John", 36)
print(p1)
```

Educator Insights: A Journal of Teaching Theory and Practice
Volume 01, Issue 05, May, 2025
brightmindpublishing.com
ISSN (E): 3061-6964

1-masala To convert the line literally, write a Python class. Rotate the line word for word.

Dastur codes	**Program Output**

class py_solution:
def reverse_words(self, s):
return ' '.join(reversed(s.split())) print(py_solution().reverse_words('P ython dasturlash tili'))
	Python programming language

2-masala Write a Python class that has the Get_String and print_String methods. get_String accepts a line from the user and prints print_String line in uppercase letter.

Dastur codes	**Dastur natijasi**

class IOString():
def init (self): self.str1 = ""
def get_String(self):
self.str1 =
input()
def print_String(self):
print(self.str1.upper())
str1 = IOString() str1.get_String()
str1.print_String()
	PYTHON PROGRAMMING LANGUAGE

3-masala Write a Python class called Rectangle structured in width and height, and a program for calculating the surface of a rectangle.

Dastur codes	**Program Output**

class Rectangle():
def init (self, l, w): self.length = l self.width = w
def rectangle_area(self): return
self.length*self.width
newRectangle = Rectangle(18, 12) print(newRectangle.rectangle_area())
	216

4-masala Write a Python class called Circle, a program for calculating the area and length of the circle, built by radius.

Dastur codes	Dastur natijasi

class Circle():
def init (self,
r):
self.radius = r
def area(self): return
self.radius**2*3.14
def length(self): return
2*self.radius*3.14
NewCircle = Circle(10) print(NewCircle.area())
print(NewCircle.length())
	314.0
	62.800000000000004

5-masala Check the examples and subclasses in a given class. Write Python program to create two blank classrooms, Student and Marks. Also check if these classes are subclasses of the object class in which they are installed.

Dastur codes	Dastur natijasi

class Student: pass
class Marks: pass
student1 = Student() marks1 = Marks()
print(isinstance(student1, Student)) print(isinstance(marks1, Student)) print(isinstance(marks1, Marks))
print(isinstance(student1, Marks))
print("Sinfning sinfostilarini
tekshiring."print(issubclass(Student, object))
print(issubclass(Marks,
object))
	True False True False
	Sinfning
	Check out the sinfosti.
	True True

RESULTS ANALYSIS

During this study, the fundamental features of classes in the Python programming language, their working principles and their application in various fields were investigated in depth. The results further confirm how important and effective classes are in the programming process. Classes allow you to organize and

Educator Insights: A Journal of Teaching Theory and Practice
Volume 01, Issue 05, May, 2025
brightmindpublishing.com
ISSN (E): 3061-6964

modulate code. Throughout the study, it was observed that the classes combined data (attributes) and behaviors (techniques) into a single whole, allowing the program code to be broken down into logical blocks. This improves the readability of the code, makes it easier to understand, and makes it easier to find errors. For example, the "Student" class contains all the information about the student and methods of working with them, which works in isolation from the other parts of the program.

Objects are the representation of classes in practice. In the process of research, it was found that classes are just "molds", while objects that work with real data and behaviors are created on the basis of these patterns. Each object is a unique instance of a class and has its own attribute values. For example, from the "Student" class, it is possible to create several student objects, each with its own name, surname and other data. Inheritance provides the ability to reuse and extend the code. In the examples studied, it was confirmed that the succession mechanism allows code to be reused by transferring the properties and methods of existing classes to new classes. This helps programmers avoid duplicate code writing and extend the functionality that is available. For example, the child classes such as "Bachelor" and "Master" may inherit the common characteristics of the "Student" parent class and have specific attributes and methods.

Polymorphism makes it possible to work with objects of different classes through a common interface. During the study, it was observed that the principle of polymorphism ensures that objects of different classes respond differently to methods of the same name. This increases the flexibility of the software and allows for overall handling of different objects. For example, for different shapes (circle, rectangle, triangle), the method of "face counting" may have the same name, but implements a specific computational algorithm for each shape. Incapulation allows for data protection and obscuring the internal structure of a class. In the case studies studied, encapsulation was confirmed to prevent random data change by limiting direct access to attributes and referring to them through methods. This reduces the risk of external code breaches when changing the internal structure of the class. These results show how powerful and flexible classes are in the Python programming language. The proper application of the classroom concept greatly simplifies the software development process, improves code quality, and ensures the long-term sustainability of projects.

Educator Insights: A Journal of Teaching Theory and Practice
Volume 01, Issue 05, May, 2025
brightmindpublishing.com
ISSN (E): 3061-6964
Licensed under CC BY 4.0 a Creative Commons Attribution 4.0 International License.

CONCLUSION

In conclusion, classes in the Python programming language are the fundamental foundation of object-oriented programming and play an important role in software design and development. In this study, the main properties of classes such as attributes and methods, as well as the processes of their creation and object instantiation were investigated in detail. Important principles of OYD such as inheritance, polymorphism, and encapsulation are also analyzed, showing their importance in reusing application code, increasing flexibility, and protecting data. Through various practical examples, the effective application of classrooms in a wide range of fields such as creating data structures, designing graphical interfaces, web programming, game creation, and scientific computing has been demonstrated. The results obtained show that the correct and appropriate use of classes significantly improves the quality of the application code, making it organized, easily understandable and reusable. The dynamic nature of Python and its full support for OYD principles provide programmers with powerful tools to efficiently manage complex software projects and create innovative solutions. In the future, a deeper study of this topic, advanced classroom opportunities and research into design concepts will serve to further increase knowledge and skills in the field of software development.

References

1. Ramalho, L. (2015). Fluent Python: Accurate, Concise, and Efficient Programming.
2. Beazley, D., & Jones, B. K. (2013-yil). Python pazandalik kitobi.
3. Zelle, J. M. (2004). Python Programming: An Introduction to Computer Science.
4. Sweigart, A. (2015). Automating Boring Jobs with Python: Complete Beginner Practical Programming.
5. Summerfield, M. (2008). Programming in Python 3: A Complete Introduction to the Python language.
6. Kurbanov, D. Q. (2017). Python Programming Fundamentals.
7. Mahmudov, N. R. (2024). Python Programming Language: Practice Training.