Python OOP

This Python OOP explains to you the Python object-oriented programming clearly so that you can apply it to develop software more effectively.

By the end of this Python OOP module, you'll have good knowledge of object-oriented principles. And you'll know how to use Python syntax to create reliable and robust software applications.

What you'll learn

- Create objects in Python by defining classes and methods.
- Extend classes using inheritance.
- SOLID principles in object-oriented programming.

Who this tutorial is for?

If you're new to object-oriented programming, or if you have basic Python skills and wish to learn in-depth how and when to correctly apply OOP in Python, this is the tutorial for you.

Section 1. Classes and objects

- Object-oriented programming (https://www.pythontutorial.net/python-oop/python-object-oriented-programming/) – introduce to you the important concepts in Python object-oriented programming.
- Class (https://www.pythontutorial.net/python-oop/python-class/) – learn how to define a class and create new objects from the class.
- Class variables (https://www.pythontutorial.net/python-oop/python-class-variables/) – explain the class variables (or attributes)
- Instance methods (https://www.pythontutorial.net/python-oop/python-methods/) – guide you on instance methods and help you understand the differences between a function and a method.
- __init__() (https://www.pythontutorial.net/python-oop/python-__init__/) – show you how to use the __init__ method to initialize object's attributes.
- Instance variables (https://www.pythontutorial.net/python-oop/python-instance-variables/) – understand the instance variables.
- Private attributes (https://www.pythontutorial.net/python-oop/python-private-attributes/) – learn about private attributes and how to use them effectively.
- Class attributes (https://www.pythontutorial.net/python-oop/python-class-attributes/) – understand class attributes and more importantly when you should use class attributes.
- Static methods (https://www.pythontutorial.net/python-oop/python-static-methods/) – explain to you static methods and shows you how to use them to group related functions in a class.

Section 2. Special methods

- __str__ method (https://www.pythontutorial.net/python-oop/python-__str__/) – show you how to use the __str__ dunder method to return the string representation of an object.
- __repr__ method (https://www.pythontutorial.net/python-oop/python-__repr__/) – learn how to use the __repr__ method and the main difference between __str__ and __repr__ methods.
- __eq__ method (https://www.pythontutorial.net/python-__eq__/) – learn how to define the equality logic for comparing objects by values.

- __hash__ method (https://www.pythontutorial.net/python-oop/python-__hash__/) – show you how to make a class hashable using the __has__ method.
- __bool__ method (https://www.pythontutorial.net/python-oop/python-__bool__/) – guide you on how to determine whether a custom object is True or False using the __bool__ method.
- __del__ method (https://www.pythontutorial.net/python-oop/python-__del__/) – understand how the __del__ method works.

Section 3. Property

- Property (https://www.pythontutorial.net/python-oop/python-properties/) – show you how to use the property class to create a property.
- @property decorator (https://www.pythontutorial.net/python-oop/python-property-decorator/) – learn how to use the @property decorator to create a property.
- Read-only property (https://www.pythontutorial.net/python-oop/python-readonly-property/) – learn how to define read-only properties and use them for computed properties.
- Delete a property (https://www.pythontutorial.net/python-oop/python-delete-property/) – guide you on how to delete a property from an object.

Section 4. Single inheritance

- Inheritance (https://www.pythontutorial.net/python-oop/python-inheritance/) – explain to you the inheritance concept and how to define a class that inherits from another class.
- Overriding methods (https://www.pythontutorial.net/python-oop/python-overriding-method/) – show you how overriding methods work.
- super() (https://www.pythontutorial.net/python-oop/python-super/) – learn how to delegate to the methods of the parent class from a method in the child class.
- __slots__ (https://www.pythontutorial.net/python-oop/python-__slots__/) – use __slots__ to make the class more memory efficient.

- Abstract class (https://www.pythontutorial.net/python-oop/python-abstract-class/) – learn what abstract classes are and how to define abstract classes.

Section 5. Enumeration

- Enumeration (https://www.pythontutorial.net/python-oop/python-enumeration/) – show you how to define a enumeration in Python.
- Enum Aliases & @enum.unique (https://www.pythontutorial.net/python-oop/python-enum-unique/) – introduce to you the enum aliases and how to use the enum.unique decorator to ensure the uniqueness of member values.
- Customize and extend enumerations (https://www.pythontutorial.net/python-oop/python-enum-class/) – learn how to customize the behaviors of enum classes and how to extend the a custom enum class.
- auto (https://www.pythontutorial.net/python-oop/python-enum-auto/) – use the enum `auto` class to generate unique values for enumeration's members.

Section 6. SOLID principles

This section introduces you to the SOLID principles of object-oriented programming. These five design principles make your software designs more maintainable and flexible.

- Single Responsibility Principle (https://www.pythontutorial.net/python-oop/python-single-responsibility-principle/) – a class should have a single resposiblity.
- Open/Closed Principle (https://www.pythontutorial.net/python-oop/python-open-closed-principle/) – a class shoud be open for extension but closed for modification.
- Liskov Substitution Principle (https://www.pythontutorial.net/python-oop/python-liskov-substitution-principle/) – a child class must be able to substituteable for its parent class.
- Interface Segregation Principle (https://www.pythontutorial.net/python-oop/python-interface-segregation-principle/) – use many small interfaces instead of a big interface.

- Dependency Inversion Principle (https://www.pythontutorial.net/python-oop/python-dependency-inversion-principle/) – make the high-level module dependent on abstraction, not the low-level module.

Section 7. Multiple inheritance

- Multiple inheritance (https://www.pythontutorial.net/python-oop/python-multiple-inheritance/) – learn how to implement multiple inheritance and understand how the method resolution order (MRO) works in Python.
- Mixin (https://www.pythontutorial.net/python-oop/python-mixin/) – introduce to you the mixin concept and how to implement mixin in Python.

Section 8. Descriptors

- Descriptors (https://www.pythontutorial.net/python-oop/python-descriptors/) – explain how descriptors work and how to use them to make the code reusable.
- Data vs non-data descriptors (https://www.pythontutorial.net/python-oop/python-data-descriptors/) – understand the differences between data and non-data descriptors

Section 9. Metaprogramming

- __new__ (https://www.pythontutorial.net/python-oop/python-__new__/) – learn how Python uses the __new__ method to create a new instance of a class.
- type class (https://www.pythontutorial.net/python-oop/python-type-class/) – show you how to dynamically create a class using the type class.
- Metaclass (https://www.pythontutorial.net/python-oop/python-metaclass/) – explain the metaclass and show you how to define a custom metaclass to create other classes.

- Metaclass example (https://www.pythontutorial.net/python-oop/python-metaclass-example/) – show you a metaclass example that allows you to inject many functionalities into classes.
- dataclass (https://www.pythontutorial.net/python-oop/python-dataclass/) – leverage dataclass to add special methods such as __init__ and __repr__ to custom classes.

Section 10. Exceptions

- Exceptions (https://www.pythontutorial.net/python-oop/python-exceptions/) – learn about exceptions in the context of objects
- Exception Handling (https://www.pythontutorial.net/python-oop/python-exception-handling/) – guide you on how to handle exceptions in the right way using the try statement.
- Raise Exceptions (https://www.pythontutorial.net/python-oop/python-raise-exception/) – show you how to use the raise statement to raise exceptions.
- Raise Exception from cause (https://www.pythontutorial.net/python-oop/python-raise-from/) – learn how to modify and forward an existing exception with a cause.
- Custom exceptions (https://www.pythontutorial.net/python-oop/python-custom-exception/) – walk you through the steps of creating a custom exception class.