

# Object-Oriented Programming

---

## Textbook

---

# Object-Oriented Programming



## Overview

Object Oriented Programming enables us to create more complex data structures that help us track information more easily. While lists, strings, tuples, arrays, and dictionaries are helpful and useful, sometimes they aren't enough.

Why wouldn't simple data types be enough? In the real world, most entities are a combination of different data types. Think about information about a friend. You could have their name (a string), their age (an integer), and hasDog (boolean) if they have a dog or not.

This is where object-oriented programming comes into play. Instead of a simple variable that holds one type of data, we can create an object that allows for multiple parts of information to exist that are all connected to the object.

You can even add functionality into your objects such as `isInvited()` as an invitation to a party that will send an invite. Object oriented programming allows for more useful applications in the real world.

Let's learn how to create objects!

# Classes

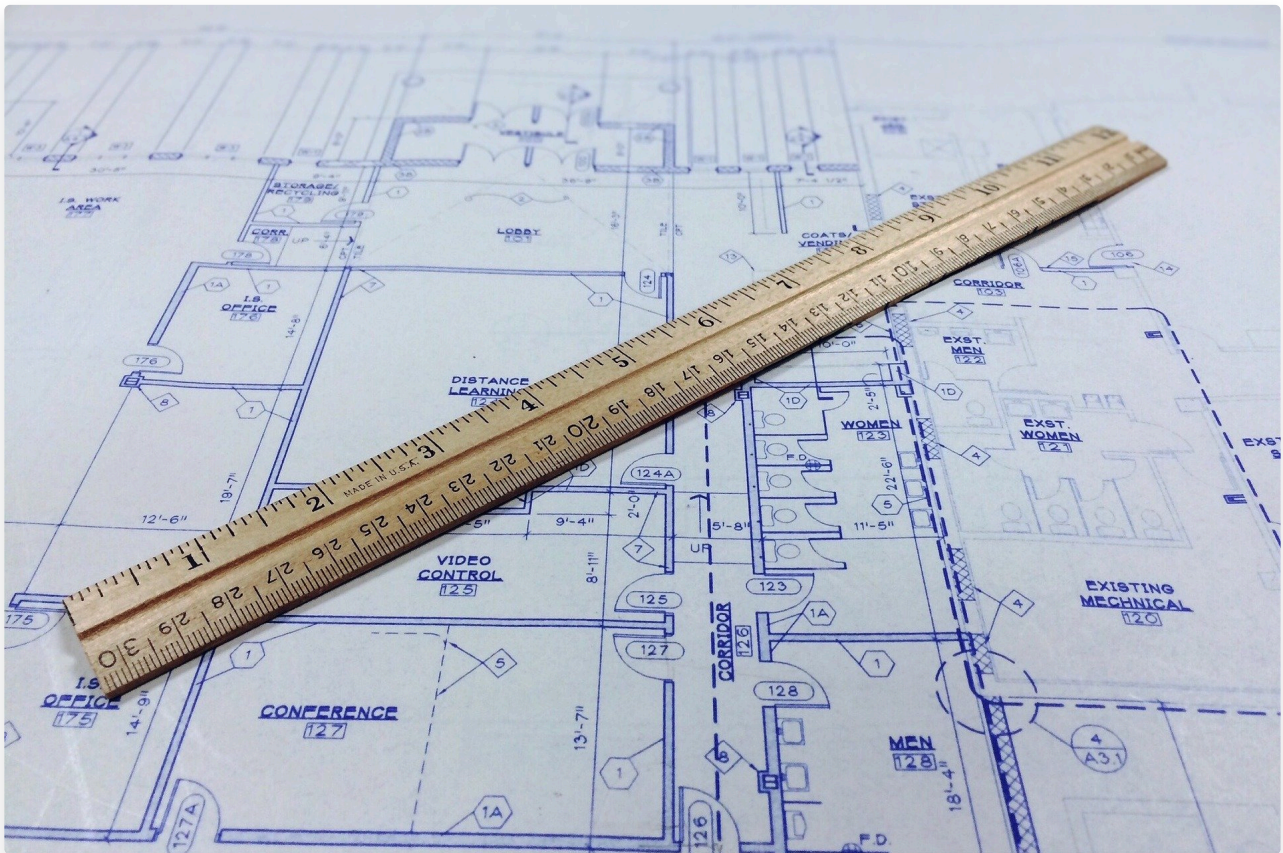
When we use object-oriented programming we create objects called classes. A class is similar to a list or a dictionary, but we can customize it to track the exact information we want to use..

In this example we're going to create a `Person` class that allows us to track the `name` , `age` , and `gender` of a person.

A class is a blueprint of an object. In our `Person` example, the class is the outline of the person and says that every person created using this class should have a `name` , `age` , and `gender` . The class specifies what specific information is needed about the person, but it doesn't actually create anything in our program.

## Creating a Class

A **class** is a blueprint of an object. It's not actually the object itself, it's just the form we use to create the object.



Just like a blueprint isn't actually a house, a **class** isn't actually an object.

Let's get started with creating an example **class**. We'll create a **class** called `Person` for this example.

To create the class, we use the `class` keyword and `Person` is the name of the class. That's always the first line of code in a class. Notice that when we create classes, they should be capitalized.

```
1 class Person:
```

Then, underneath that class declaration we put a function that is always the same, no matter what type of class you're creating. The function is `__init__()` which stands for **initialize**. This function is used to *initialize* the class when we create our first object.

```
def __init__( )
```



NOTICE! There are TWO underscores before the `__init__` and TWO underscores after the `__init__` in this code. There is also a **space** between the `def` and the first underscore! It must be typed this way to work correctly. This is correct Python syntax.

In the `__init__()` function we need to add the parameters. The `__init__()` function MUST always have a `self` parameter. In addition to the `self` parameter, we add in any attributes we want our class to have. In this case, our `Person` class will have `name`, `age`, and `gender` attributes, so these are added in as parameters.

```
1 class Person:
2
3     def __init__(self, name, age, gender):
```

The final step in creating this function is to add in the attribute assignments. This is where we set the name of the `__class__` attribute to the name of the person we're creating. See the example below:

```
1 class Person:
2
3     def __init__(self, name, age, gender):
4         self.name = name
5         self.age = age
6         self.gender = gender
7
```

There always needs to be one of these assignments for each attribute of the class.

So now we have just created a class! Remember that this class is not an object!! This class is a template we use to create an object. Another way to say creating an object using a class is to say we create an **instance** of the class.

## Instances

Think about the last time you had to fill out a form about yourself. The form you filled out is exactly the same form that was given to everyone else – this is like a class. When you wrote your specific information in the form, that had information that was specific to you. So your information is like an **instance** of the class.

Instances are actual objects created using the class. When we add a person to our program using the class, the instance will be the actual person we created.

For example, let's say we want to add our friend Jasmine. When we create Jasmine in our program, she will be an instance of the person class. Her `name` attribute will be "Jasmine", her `age` will be "15" and her `gender` will be "female".

Your personal, filled-out form is an instance of the class template.

We will learn more about creating instances of classes (or objects) in the next lesson.

## Checkpoint

---

### Object Oriented Programming

Create a class!

Practice creating a class—or a template for an object. Remember that you aren't actually creating an object yet, but a template you can use to create an object in the next lesson.

1. Create a class named `Dog` .
2. Inside the class, include three attributes: `name` , `color` , `size`
3. Make sure the attributes are in that order.

## Requirements:

- Create a class named `Dog`
- Inside the class named `Dog` , create the `__init__( )` section
- Reference the `name` attribute to the class name using `self` .
- Reference the `color` attribute to the class name using `self` .
- Reference the `size` attribute to the class name using `self` .

## Questions (10)

### 1. What is this code an example of?

MULTIPLE CHOICE

```
class Fish: def __init__(self, color, type, size): self.color = color self.type = type self.size = size
```

Choose the correct answer:

- A. A class
- B. An instance
- C. An object
- D. A fish

### 2. True or False: A class is another name for an object

MULTIPLE CHOICE

Choose the correct answer:

- A. True
- B. False

### 3. Which of the following would be the correct way to create a class?

MULTIPLE CHOICE

Choose the correct answer:

- A. class music:
- B. class Music:
- C. classMusic:
- D. classmusic:

**4. True or False: Create an instance of a 'class' is another way of saying 'create an object' using the class**

MULTIPLE CHOICE

**Choose the correct answer:**

- A. True
- B. False

**5. In OOP, what do we call the part of the code that acts like a blueprint?**

MULTIPLE CHOICE

**Choose the correct answer:**

- A. class
- B. object
- C. instance
- D. iteration

**6. What is the purpose of the `__init__()` function in a class?**

MULTIPLE CHOICE

**Choose the correct answer:**

- A. To create an object
- B. To define the class structure
- C. To create a new instance
- D. To execute the main program

**7. Which keyword is used to declare a class in Python?**

MULTIPLE CHOICE

**Choose the correct answer:**

- A. define
- B. class
- C. create
- D. initiate

## 8. What is an "instance" in the context of OOP?

MULTIPLE CHOICE

Choose the correct answer:

- A. A synonym for a class
- B. A blueprint for creating objects
- C. A specific object created from a class
- D. A method to define class attributes

## 9. Debug the following code:

DEBUG CODE

Code to Debug:

```
1 class Dog:
2
3     def __init__(self, name, age, gender)
4         self.name = name
5         self.age = age
6         self.gender = gender
```

## 10. Debug the following code:

DEBUG CODE

Code to Debug:

```
1 class dog:
2
3     def __init__(self, name, age, gender):
4         self.name = name
5         self.age = age
6         self.gender = gender
```

## Challenges (3)

### 1. Antique Car Show

Create a class for an antique car show!

Practice creating a class—or a template for an object. Remember that you aren't actually creating an object yet, but a template you can use to create an object in the next lesson.

1. Create a class named `Car` .
2. Inside the class, include three attributes: `kind` , `color` , `year`
3. Make sure the attributes are in that order.

#### Requirements:

- Create a class named `Car`
- Inside the class named `Car` , create the `__init__( )` section
- Reference the `kind` attribute to the class name using `self` .
- Reference the `color` attribute to the class name using `self` .
- Reference the `year` attribute to the class name using `self` .

### 2. Arborist

Did you know that someone who studies and works with trees is called an Arborist?

Create a class that can be used to create objects about trees!

Practice creating a class—or a template for an object. Remember that you aren't actually creating an object yet, but a template you can use to create an object in the next lesson.

1. Create a class named `Tree` .
2. Inside the class, include five attributes: `fruit` , `height` , `leaf` , `flower` , `seed`
3. Make sure the attributes are in that order.

#### Requirements:

- Create a class named `Tree`
- Inside the class named `Tree` , create the `__init__( )` section
- Reference the `fruit` attribute to the class name using `self` .
- Reference the `height` attribute to the class name using `self` .
- Reference the `leaf` attribute to the class name using `self` .
- Reference the `flower` attribute to the class name using `self` .
- Reference the `seed` attribute to the class name using `self` .

### 3. Bake Sale

You are hosting a bake sale to raise money for a charity you want to support. You want people to make many kinds of desserts but use the same general pattern. So you decide to create a class.

Practice creating a class—or a template for an object. Remember that you aren't actually creating an object yet, but a template you can use to create an object in the next lesson.

1. Create a class named `Dessert` .
2. Inside the class, include four attributes: `shape` , `size` , `filling` , `topping`
3. Make sure the attributes are in that order.

#### Requirements:

- Create a class named `Dessert`
- Inside the class named `Dessert` , create the `__init__( )` section. Include the attributes.
- Reference the `shape` attribute to the class name using `self` .
- Reference the `size` attribute to the class name using `self` .
- Reference the `filling` attribute to the class name using `self` .
- Reference the `topping` attribute to the class name using `self` .



## Answer Keys & Solutions

### Checkpoint Solutions

#### Object Oriented Programming

```
1 class Dog:
2
3     def __init__(self, name, color, size):
4         self.name = name
5         self.color = color
6         self.size = size
```

### Questions

#### 1. What is this code an example of?

MULTIPLE CHOICE

Correct Answer:

- A. A class ✓ Correct
- B. An instance ✗ Incorrect
- C. An object ✗ Incorrect
- D. A fish ✗ Incorrect

#### Explanation:

This is the template used to create objects.

#### 2. True or False: A class is another name for an object

MULTIPLE CHOICE

Correct Answer:

- A. True ✗ Incorrect
- B. False ✓ Correct

#### Explanation:

A class is used to create an object.

### 3. Which of the following would be the correct way to create a class?

MULTIPLE CHOICE

**Correct Answer:**

- A. class music: ✗ Incorrect
- B. class Music: ✓ Correct
- C. classMusic: ✗ Incorrect
- D. classmusic: ✗ Incorrect

#### **Explanation:**

Classes are capitalized.

### 4. True or False: Create an instance of a 'class' is another way of saying 'create an object' using the class

MULTIPLE CHOICE

**Correct Answer:**

- A. True ✓ Correct
- B. False ✗ Incorrect

#### **Explanation:**

Objects and instances of classes are the same thing.

### 5. In OOP, what do we call the part of the code that acts like a blueprint?

MULTIPLE CHOICE

**Correct Answer:**

- A. class ✓ Correct
- B. object ✗ Incorrect
- C. instance ✗ Incorrect
- D. iteration ✗ Incorrect

#### **Explanation:**

The class is used to create objects

## 6. What is the purpose of the `__init__()` function in a class?

MULTIPLE CHOICE

**Correct Answer:**

- A. To create an object ✗ Incorrect
- B. To define the class structure ✓ Correct
- C. To create a new instance ✗ Incorrect
- D. To execute the main program ✗ Incorrect

### Explanation:

The `__init__()` function creates the blueprint of an object

## 7. Which keyword is used to declare a class in Python?

MULTIPLE CHOICE

**Correct Answer:**

- A. define ✗ Incorrect
- B. class ✓ Correct
- C. create ✗ Incorrect
- D. initiate ✗ Incorrect

### Explanation:

The keyword matches the goal

## 8. What is an "instance" in the context of OOP?

MULTIPLE CHOICE

**Correct Answer:**

- A. A synonym for a class ✗ Incorrect
- B. A blueprint for creating objects ✗ Incorrect
- C. A specific object created from a class ✓ Correct

D. A method to define class attributes

✖ Incorrect

### Explanation:

The instance of a class is what we call an object

## 9. Debug the following code:

DEBUG CODE

### Incorrect Code:

```
1 class Dog:
2
3     def __init__(self, name, age, gender)
4         self.name = name
5         self.age = age
6         self.gender = gender
```

### Correct Solution:

```
1 class Dog:
2
3     def __init__(self, name, age, gender):
4         self.name = name
5         self.age = age
6         self.gender = gender
```

### Explanation:

This code is missing a colon

## 10. Debug the following code:

DEBUG CODE

### Incorrect Code:

```
1 class dog:
2
3     def __init__(self, name, age, gender):
4         self.name = name
5         self.age = age
6         self.gender = gender
```

### Correct Solution:

```
1 class Dog:
2
3     def __init__(self, name, age, gender):
4         self.name = name
5         self.age = age
6         self.gender = gender
```

**Explanation:**

The class name needs to be capitalized

## Challenges

### 1. Antique Car Show

**Solution:**

```
1 class Car:
2
3     def __init__(self, kind, color, year):
4         self.kind = kind
5         self.color = color
6         self.year = year
```

### 2. Arborist

**Solution:**

```
1 class Tree:
2
3     def __init__(self, fruit, height, leaf, flower, seed):
4         self.fruit = fruit
5         self.height = height
6         self.leaf = leaf
7         self.flower = flower
8         self.seed = seed
```

### 3. Bake Sale

**Solution:**

```
1 class Dessert:
2
3     def __init__(self, shape, size, filling, topping):
4         self.shape = shape
5         self.size = size
6         self.filling = filling
7         self.topping = topping
```