

Creating an Instance/Object

Textbook

Creating an Instance/Object



Overview

To review - a **class** is a blueprint of an object. An instance is the implementation of that blueprint to create an actual object in our program.

For example, let's say we used the same blueprints to build five different houses. Each of these houses has different owners, and they were able to customize the materials in each of their houses. Even though they're built from the same blueprints, each house still has unique attributes.

In this example, each of the five different houses is an *instance* of the house **class**. The blueprints are the exact same, but each *instance* has some different attributes.

Create an Object/Instance

Now that we have created a **Dog** class, we're now able to create an instance of that class, or in other words, we are ready to create an object!

Let's create an instance of a dog named Jasmine:

```

1 class Dog:
2
3     def __init__(self, name, age, gender):
4         self.name = name
5         self.age = age
6         self.gender = gender
7
8 pet = Dog("Jasmine", "15", "Female")
9
10 print(pet)

```

Try it!

As you can see, we created a new variable named `pet` and set that equal to a new instance of the `Dog` class. The variable named `pet` is an object that represents Jasmine in our program. The `pet` object has the attributes of `Jasmine` as the name, `15` as the age, and `Female` as the gender.

To be very clear: The object named `pet` looks like this.

```
pet = Dog("Jasmine", "15", "Female")
```

This object is built using the class named `Dog`.

Accessing Attributes of an Object

As you can see from the above example, when you print `pet`, it returns a reference to that object. It doesn't return the attributes of your object. Let's learn how to access the specific attributes of your object named `pet`.

This is done by calling the object name and the attribute like this:

```
pet.attribute
```

```

1 class Dog:
2
3     def __init__(self, name, age, gender):
4         self.name = name
5         self.age = age
6         self.gender = gender
7
8 pet = Dog("Jasmine", "15", "Female")
9
10 print(pet.name)

```

Try it!

We can also concatenate different parts of the object like this.

```

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

```

```
8 pet = Dog("Jasmine", "15", "Female")
9
10 print("Pet 1: " + pet.name + " " + pet.age + " " + pet.gender)
```

Try it!

Another Instance of the Class named Dog

Let's create another pet, a young dog named `George`. We will use the class named `Dog` to create another object, or another instance of the class.

```
1 class Dog:
2
3     def __init__(self, name, age, gender):
4         self.name = name
5         self.age = age
6         self.gender = gender
7
8 pet1 = Dog("Jasmine", "15", "Female")
9 pet2 = Dog("George", "3", "Male")
10
11 print("Pet 1: " + pet1.name + " " + pet1.age + " " + pet1.gender)
12 print("Pet 2: " + pet2.name + " " + pet2.age + " " + pet2.gender)
```

Try it!

Now, if we want to access the information about each of these objects, it's very easy. See the sample code below, along with what that code would print out:

```
1 class Dog:
2
3     def __init__(self, name, age, gender):
4         self.name = name
5         self.age = age
6         self.gender = gender
7
8 pet1 = Dog("Jasmine", "15", "Female")
9 pet2 = Dog("George", "3", "Male")
10
11 print(pet1.name)
12 print(pet1.gender)
13 print(pet1.age)
14 print(pet2.gender)
15 print(pet2.name)
```

Try it!

Create your own Instance

Following the steps above, add onto the `Pet` class you made in the previous project. Create a new instance of a `Pet`, give it a `name` and a `species`, and print it out.

Checkpoint

Create an Instance

Create an instance of a class!

1. Create a class named `Hat`.
2. Inside the class, include three attributes: `kind`, `color`, `material`.
3. Make sure the attributes are in that order.
4. Create an instance of the class named `Hat`. Name the instance `myObject`.

Requirements:

- Create a class named `Hat`.
- Inside the class named `Hat`, 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 `material` attribute to the class name using `self`.
- Create an object named `myObject` by using the `Hat` template.

Questions (10)

1. What is the following code an example of?

MULTIPLE CHOICE

```
first = Animal("Furry", "Brown", "4 legs")
```

Choose the correct answer:

- A. An object
- B. A class
- C. A list
- D. A loop

2. What is another way of saying "create an instance of a class?"

MULTIPLE CHOICE

Choose the correct answer:

- A. Create an object
- B. Create a template or blueprint
- C. Create a list
- D. Create a variable

MULTIPLE CHOICE

3. Which of the following is most similar to a blueprint?

Choose the correct answer:

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

4. Which of the following is the way to access the attribute "salad" in the object named myObject?

MULTIPLE CHOICE

```
class Dinner: def __init__(self, main, side, drink): self.main = main self.side = side self.drink = drink myObject = Dinner("pasta", "salad", "juice") print(myObject)
```

Choose the correct answer:

- A. myObject.side
- B. myObject.salad
- C. Dinner.side
- D. Dinner.salad

MULTIPLE CHOICE

5. How many objects can be made with one class?

Choose the correct answer:

- A. One
- B. As many objects as there are attributes in the class.
- C. As many as you want.
- D. 4

MULTIPLE CHOICE

6. Which term is used to describe an actual object created from a class blueprint?

Choose the correct answer:

- A. Blueprint
- B. Implementation
- C. Instance
- D. Reference

7. How do you access the attributes of an object named p1?

Choose the correct answer:

- A. p1.attribute
- B. attribute.p1
- C. p1[attribute]
- D. p1->attribute

8. What does the following code print?

```
class Person: def __init__(self, name, age, gender): self.name = name self.age = age self.gender = gender p1 = Person("George", "67", "Male") print("Person 1: " + p1.name + " " + p1.age + " " + p1.gender)
```

Choose the correct answer:

- A. Person 1: George 67 Male
- B. Person 1: 67 Male George
- C. George 67 Male
- D. Person 1: "George 67 Male"

9. What is the purpose of creating multiple instances of a class?

Choose the correct answer:

- A. To create different classes
- B. To customize attributes for each instance
- C. To share attributes among instances
- D. To make the code longer

10. Debug the following code:

Code to Debug:

```
1 class Person:
2
3     def __init__(self, name, age, gender):
4         self.name = name
5         self.age = age
6         self.gender = gender
7
8 p1 = person("Jasmine", "15", "Female")
9
10 print(p1.name)
```

Challenges (3)

1. A Fruit Festival

You are building an app for a fruit arrangement company. They want to have a template that would be easy to create different bouquets.

1. Create 4 different objects using the same class!
2. Use the following class named `Fruit` to create your objects.
3. Print the `kind` attribute for each of your objects.

Example Output:

```
class Fruit: def __init__(self, shape, kind, size): self.shape = shape self.kind = kind self.size = size
```

Requirements:

- Create 4 objects using the class named `Fruit`.
- Print the `kind` attribute for each of your objects.

2. Pet Store

Create a class named `Pet` with at least 3 attributes

Use the class named `Pet` to create at least 3 instances of the class.

Requirements:

- Create a class named `Pet`
- Inside the class named `Pet`, create the `__init__()` section
- Reference all 3 attributes to the class name using `self`.
- Create 3 instances of the object named `Pet`

3. Vacation Planner

1. Create a class named `Vacation` with at least 3 attributes: `location` , `activity` , `food` (in that order)
2. Use the class named `Vacation` to create at least 3 instances of the class.
3. Use an attribute from each instance in a concatenated sentence.

Requirements:

- Create a class named `Vacation`
- Inside the class named `Vacation` , create the `__init__()` section with the designated attributes in the correct order.
- Reference the `location` attribute to the class name using `self` .
- Reference the `activity` attribute to the class name using `self` .
- Reference the `food` attribute to the class name using `self` .
- Create three objects by using the `Vacation` template.
- Use an attribute from each instance in 3 separate concatenated sentences.

Answer Keys & Solutions

Checkpoint Solutions

Create an Instance

```
1 class Hat:
2
3     def __init__(self, kind, color, material):
4         self.kind = kind
5         self.color = color
6         self.material = material
7
8 myObject = Hat("warm fuzzy", "pink", "fleece")
9
10 print(myObject)
```

Questions

1. What is the following code an example of?

MULTIPLE CHOICE

Correct Answer:

- | | |
|--------------|-------------|
| A. An object | ✓ Correct |
| B. A class | ✗ Incorrect |
| C. A list | ✗ Incorrect |
| D. A loop | ✗ Incorrect |

Explanation:

This bit of code is using a class named Animal to create something.

2. What is another way of saying "create an instance of a class?"

MULTIPLE CHOICE

Correct Answer:

- | | |
|-----------------------------------|-------------|
| A. Create an object | ✓ Correct |
| B. Create a template or blueprint | ✗ Incorrect |
| C. Create a list | ✗ Incorrect |

D. Create a variable

✗ Incorrect

Explanation:

This is how to use object oriented programming.

3. Which of the following is most similar to a blueprint?

MULTIPLE CHOICE

Correct Answer:

A. class

✓ Correct

B. object

✗ Incorrect

C. instance

✗ Incorrect

D. variable

✗ Incorrect

Explanation:

This helps to create objects.

4. Which of the following is the way to access the attribute "salad" in the object named myObject?

MULTIPLE CHOICE

Correct Answer:

A. myObject.side

✓ Correct

B. myObject.salad

✗ Incorrect

C. Dinner.side

✗ Incorrect

D. Dinner.salad

✗ Incorrect

Explanation:

10

5. How many objects can be made with one class?

MULTIPLE CHOICE

Correct Answer:

- A. One ✗ Incorrect
- B. As many objects as there are attributes in the class. ✗ Incorrect
- C. As many as you want. ✓ Correct
- D. 4 ✗ Incorrect

Explanation:

Remember that a class is like a blueprint for a house.

6. Which term is used to describe an actual object created from a class blueprint?

MULTIPLE CHOICE

Correct Answer:

- A. Blueprint ✗ Incorrect
- B. Implementation ✗ Incorrect
- C. Instance ✓ Correct
- D. Reference ✗ Incorrect

Explanation:

In programming, an object is called an instance of the class

7. How do you access the attributes of an object named p1?

MULTIPLE CHOICE

Correct Answer:

- A. p1.attribute ✓ Correct
- B. attribute.p1 ✗ Incorrect
- C. p1[attribute] ✗ Incorrect
- D. p1->attribute ✗ Incorrect

Explanation:

The object name comes first.

8. What does the following code print?

Correct Answer:

- A. Person 1: George 67 Male ✓ Correct
- B. Person 1: 67 Male George ✗ Incorrect
- C. George 67 Male ✗ Incorrect
- D. Person 1: "George 67 Male" ✗ Incorrect

Explanation:

The name comes first, then the age, then the gender

9. What is the purpose of creating multiple instances of a class?

Correct Answer:

- A. To create different classes ✗ Incorrect
- B. To customize attributes for each instance ✓ Correct
- C. To share attributes among instances ✗ Incorrect
- D. To make the code longer ✗ Incorrect

Explanation:

Each instance has specific attributes

10. Debug the following code:

Incorrect Code:

```
1 class Person:
2
3     def __init__(self, name, age, gender):
4         self.name = name
5         self.age = age
6         self.gender = gender
7
8 p1 = person("Jasmine", "15", "Female")
9
```

```
10 print(p1.name)
```

Correct Solution:

```
1 class Person:
2
3     def __init__(self, name, age, gender):
4         self.name = name
5         self.age = age
6         self.gender = gender
7
8 p1 = Person("Jasmine", "15", "Female")
9
10 print(p1.name)
```

Explanation:

There's a word that needs to be capitalized

Challenges

1. A Fruit Festival

Solution:

```
1 class Fruit:
2
3     def __init__(self, shape, kind, size):
4         self.shape = shape
5         self.kind = kind
6         self.size = size
7
8
9 first = Fruit("flower", "apple", "small")
10 second = Fruit("bird", "watermelon", "tall")
11 third = Fruit("pattern", "avocado", "small")
12 fourth = Fruit("bird", "orange", "big")
13
14 print(first.kind)
15 print(second.kind)
16 print(third.kind)
17 print(fourth.kind)
```

2. Pet Store

Solution:

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

```
8
9 first = Pet("Speedy", "lizard", "2")
10 second = Pet("Spot", "dog", "4")
11 third = Pet("Ferdinand", "goat", "5")
```

3. Vacation Planner

Solution:

```
1 class Vacation:
2
3     def __init__(self, location, activity, food):
4         self.location = location
5         self.activity = activity
6         self.food = food
7
8
9 first = Vacation("Beach", "snorkeling", "smoothies")
10 second = Vacation("Mountain", "rock climbing", "dutch oven")
11 third = Vacation("Theme Park", "roller coaster", "pizza")
12
13 print("I'm excited to go visit the " + first.location)
14 print("I would love to do some " + second.activity)
15 print("I love to eat " + third.food)
```