Python Tutorial
Python HOME
Python Intro
Python Get Started
Python Syntax
Python Comments
Python Variables
Python Data Types
Python Numbers
Python Casting
Python Strings
Python Booleans
Python Operators
Python Lists
Python Tuples
Python Sets
Python Dictionaries
Python If...Else
Python While Loops
Python For Loops
Python Functions
Python Lambda

BUILD YOUR CAREER. GET FULL ACCESS. SAVE 770$
Start today
ADVERTISEMENT

Python Classes and Objects

Previous Next

Python Classes/Objects

Python is an object oriented programming language.

Almost everything in Python is an object, with its properties and methods.

A Class is like an object constructor, or a "blueprint" for creating objects.

Create a Class

To create a class, use the keyword `class` :

Example

Create a class named MyClass, with a property named x:

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

Try it Yourself »

Create Object

Now we can use the class named MyClass to create objects:

Example

Create an object named p1, and print the value of x:

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

Try it Yourself »

The __init__() Function

The examples above are classes and objects in their simplest form, and are not really useful in real life applications.

To understand the meaning of classes we have to understand the built-in __init__() function.

All classes have a function called __init__(), which is always executed when the class is being initiated.

Use the __init__() function to assign values to object properties, or other operations that are necessary to do when the object is being created:

Example

Create a class named Person, use the __init__() function to assign values for name and age:

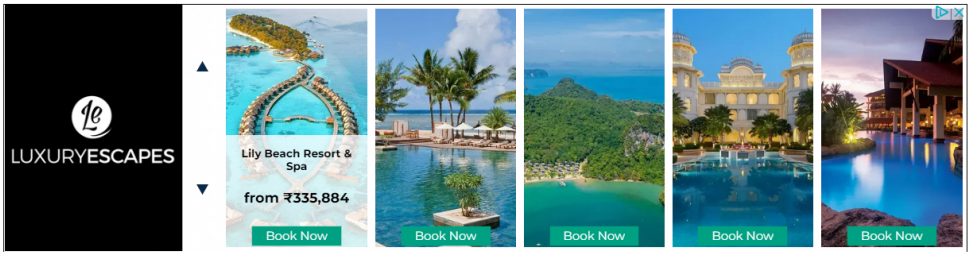
```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

p1 = Person("John", 36)

print(p1.name)
print(p1.age)
```

Visual Learner?
Be taught by an instructor through our video tutorial!
WATCH NOW
COLOR PICKER
Get certified by completing a Python course today!
Get started
IMPROVE YOUR Learning Journey
No More Ads
Hosting Benefits
More Spaces
HTML Video
UPGRADE
ADVERTISEMENT

Try it Yourself »

Note: The `__init__()` function is called automatically every time the class is being used to create a new object.

ADVERTISEMENT



Build your career. Get Full Access.

Lifelong access to all W3Schools courses

The __str__() Function

The __str__() function controls what should be returned when the class object is represented as a string.

If the __str__() function is not set, the string representation of the object is returned:

Example

The string representation of an object WITHOUT the __str__() function:

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

p1 = Person("John", 36)

print(p1)
```

Try it Yourself »

Example

The string representation of an object WITH the __str__() function:

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def __str__(self):
    return f"{self.name}({self.age})"

p1 = Person("John", 36)

print(p1)
```

Try it Yourself »

Object Methods

Objects can also contain methods. Methods in objects are functions that belong to the object.

Let us create a method in the Person class:

Example

Insert a function that prints a greeting, and execute it on the p1 object:

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def myfunc(self):
    print("Hello my name is " + self.name)
```

```
p1 = Person("John", 36)
p1.myfunc()
```

Try it Yourself »

Note: The `self` parameter is a reference to the current instance of the class, and is used to access variables that belong to the class.

The self Parameter

The `self` parameter is a reference to the current instance of the class, and is used to access variables that belongs to the class.

It does not have to be named `self` , you can call it whatever you like, but it has to be the first parameter of any function in the class:

Example

Use the words *mysillyobject* and *abc* instead of *self*:

```
class Person:
  def __init__(mysillyobject, name, age):
    mysillyobject.name = name
    mysillyobject.age = age

  def myfunc(abc):
    print("Hello my name is " + abc.name)

p1 = Person("John", 36)
p1.myfunc()
```

Try it Yourself »

Modify Object Properties

You can modify properties on objects like this:

Example

Set the age of p1 to 40:

```
p1.age = 40
```

Try it Yourself »

Delete Object Properties

You can delete properties on objects by using the `del` keyword:

Example

Delete the age property from the p1 object:

```
del p1.age
```

Try it Yourself »

Delete Objects

You can delete objects by using the `del` keyword:

Example

Delete the p1 object:

```
del p1
```

Try it Yourself »

The pass Statement

`class` definitions cannot be empty, but if you for some reason have a `class` definition with no content, put in the `pass` statement to avoid getting an error.

Example

```
class Person:
  pass
```

Try it Yourself »

Test Yourself With Exercises

Exercise:

Create a class named MyClass:

```
    MyClass:
  x = 5
```

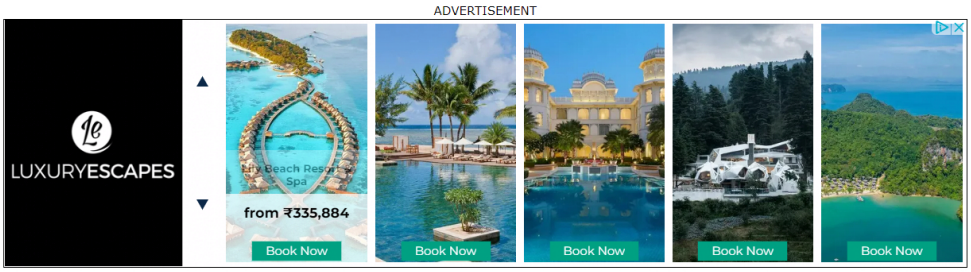
Submit Answer »

Start the Exercise

‹ Previous

Next ›

ADVERTISEMENT



ADVERTISEMENT

Java Tutorial
C++ Tutorial
jQuery Tutorial

Java Reference
Angular Reference
jQuery Reference

Java Examples
XML Examples
jQuery Examples

C++ Certificate
C# Certificate
XML Certificate

FORUM | ABOUT

W3Schools is optimized for learning and training. Examples might be simplified to improve reading and learning. Tutorials, references, and examples are constantly reviewed to avoid errors, but we cannot warrant full correctness of all content. While using W3Schools, you agree to have read and accepted our terms of use, cookie and privacy policy.

Copyright 1999-2022 by Refsnes Data. All Rights Reserved.
W3Schools is Powered by W3.CSS.