Python Classes and Objects

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
```

Run example »

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)
```

Run example »

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)
```

Run example »

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

Object Methods

Objects can also contain methods. Methods in objects are functions that belongs 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:

```python
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()
```

Run example »

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

The self Parameter

The `self` parameter is a reference to the class itself, 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*:

```python
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()
```

Run example »

Modify Object Properties

You can modify properties on objects like this:

Example

Set the age of p1 to 40:

```python
p1.age = 40
```

Run example »

Delete Object Properties

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

Example

Delete the age property from the p1 object:

```python
del p1.age
```

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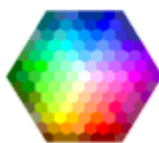
You can delete objects by using the `del` keyword:

Example

Delete the p1 object:

```
del p1
```

COLOR PICKER

HOW TO

Tabs
Dropdowns
Accordions
Side Navigation
Top Navigation
Modal Boxes
Progress Bars
Parallax
Login Form
HTML Includes

SHARE

CERTIFICATES

HTML
CSS
JavaScript
PHP
jQuery
Bootstrap
XML

Read More »

REPORT ERROR

PRINT PAGE

FORUM

ABOUT

Top 10 Tutorials

SQL Tutorial
PHP Tutorial
jQuery Tutorial
Python Tutorial

Top 10 References

HTML Reference
CSS Reference
JavaScript Reference
W3.CSS Reference
Bootstrap Reference
SQL Reference
PHP Reference
HTML Colors
jQuery Reference
Python Reference

Top 10 Examples

HTML Examples
CSS Examples
JavaScript Examples
How To Examples
W3.CSS Examples
Bootstrap Examples
PHP Examples
jQuery Examples
Angular Examples
XML Examples

Web Certificates

HTML Certificate
CSS Certificate
JavaScript Certificate
jQuery Certificate
PHP Certificate
Bootstrap Certificate
XML Certificate

w3schools.com