

Python

Classes

&

Objects

Class IX

Lab 12

**Lab Objectives:**

* What is dictionary
* Access values
* Change values
* Delete, Clear, Loop and Constructor

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:

|  |
| --- |
| demo\_class1.py:  class MyClass:  x = 5  print(MyClass) |

|  |
| --- |
| C:\Users\My Name>python demo\_class1.py  <class '\_\_main\_\_.MyClass'> |

Create Object

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

|  |
| --- |
| class MyClass:  x = 5  p1 = MyClass()  print(p1.x) |

|  |
| --- |
| Output  5 |

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:

|  |
| --- |
| class Person:  def \_\_init\_\_(self, name, age):  self.name = name  self.age = age  p1 = Person("John", 36)  print(p1.name)  print(p1.age) |

|  |
| --- |
| John 36 |

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:

|  |
| --- |
| *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()* |

|  |
| --- |
| Hello my name is John |

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:

Modify Object Properties

You can modify properties on objects like this:

|  |
| --- |
| *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.age = 40*  *print(p1.age)* |

|  |
| --- |
| 40 |

Delete Object Properties

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

**Example**

Delete the age property from 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)*  *del p1.age*  *print(p1.age)* |

|  |
| --- |
| Traceback (most recent call last):   File "demo\_class7.py", line 13, in <module>     print(p1.age) AttributeError: 'Person' object has no attribute 'age' |

Delete Objects

You can delete objects by using the del keyword:

Example

Delete 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)*  *del p1*  *print(p1)* |

|  |
| --- |
| Traceback (most recent call last):   File "demo\_class8.py", line 13, in <module>     print(p1) NameError: 'p1' is not defined |