

# 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`:

```
In [16]: # Example  
# Create a class named MyClass, with a property named x:  
  
class MyClass:  
    x = 5
```

## Create Object

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

```
In [17]: # Example  
# Create an object named p1, and print the value of x:  
  
p1 = MyClass()  
print(p1.x)
```

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:

```
In [4]: # 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)
```

John  
36

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

## 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:

```
In [6]: # 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)
```

<\_\_main\_\_.Person object at 0x000001B94D99AE20>

```
In [8]: # 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)
```

John(36)

## 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:

```
In [19]: # 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 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:

```
In [20]: # 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()
```

Hello my name is John

## Modify Object Properties

You can modify properties on objects like this:

```
In [11]: # Example
# Set the age of p1 to 40:

p1.age = 40
```

## Delete Object Properties

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

```
In [12]: # Example
# Delete the age property from the p1 object:

del p1.age
```

## Delete Objects

You can delete objects by using the del keyword:

```
In [13]: # Example
# Delete the p1 object:

del p1
```

## 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.

```
In [15]: # Example  
class Person:  
    pass
```

```
In [ ]:
```