PYTHON

CLASS AND OBJECTS

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

CREATING A CLASS

To create a class, use the keyword class:

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

CREATING A OBJECTS

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

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

OBJECTS

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:

FOR A 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)
```

OBJECTS

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:

FOR A EXAMPLE

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

p1 = Person("John", 36)

print(p1)
```

OBJECTS 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()
```