Basic Python #6

AI Mentorship

Outline

1. Python: Classes
2. Python: Objects
3. Python: The __init__() Function
4. Python: Object Methods

Python: Classes

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

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

Python: Objects

An Object is an instance of a Class. A class is like a blueprint while an instance is a copy of the class with *actual values*.

Create Object
Now we can use the class named MyClass to create objects:
Create an object named p1, and print the value of x:

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

Python: The __init__() Function

The examples before 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.

Python: The __init__() Function

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)
```

Python: Object Methods

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

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()
```