PYTHON

SECTION TEN

Introduction to Python| Objects & Classes | Pragmatic Agility

OBJECTS

- Code organizational technique
- DRY concept. Don't Repeat Yourself
- Reusable code blocks
- Organize data into complex concepts
- Defined with class definitions

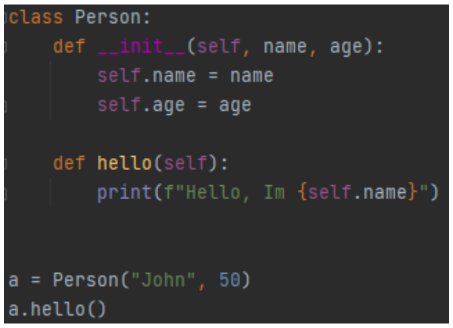
CLASS PARTS

- Class attributes
    - variables for objects
- Class methods
    - functions for objects
- Special methods are available for class objects that accomplish specific goals. __init__ runs automatically when the class is created

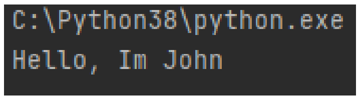
MORE INFO

- _init_ is known as a constructor in other languages
- Inheritance is another DRY technique. It shares code between classes
- Composition is when a class contains another class
- Python polymorphism is considered loose

QUICK EXAMPLES

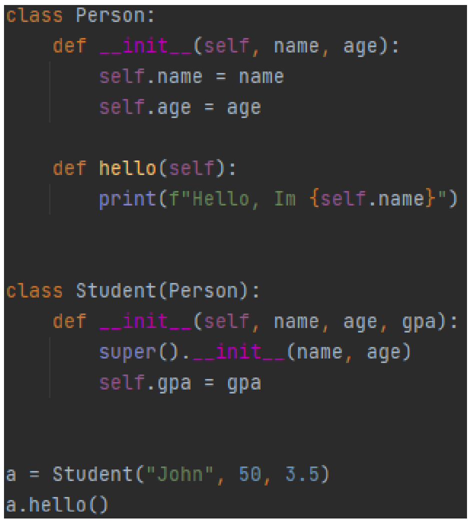


```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def hello(self):
        print(f"Hello, Im {self.name}")


a = Person("John", 50)
a.hello()
```



```
C:\Python38\python.exe
Hello, Im John
```

The Person object organizes data into a person concept. The _init_ method creates and assigns class attributes



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

    def hello(self):
        print(f"Hello, Im {self.name}")


class Student(Person):
    def __init__(self, name, age, gpa):
        super().__init__(name, age)
        self.gpa = gpa


a = Student("John", 50, 3.5)
a.hello()
```

The student reuses applicable person code. We must call the super() method to make sure the parent init method runs.