Objects and Constructors

JavaScript's equivalent to classes

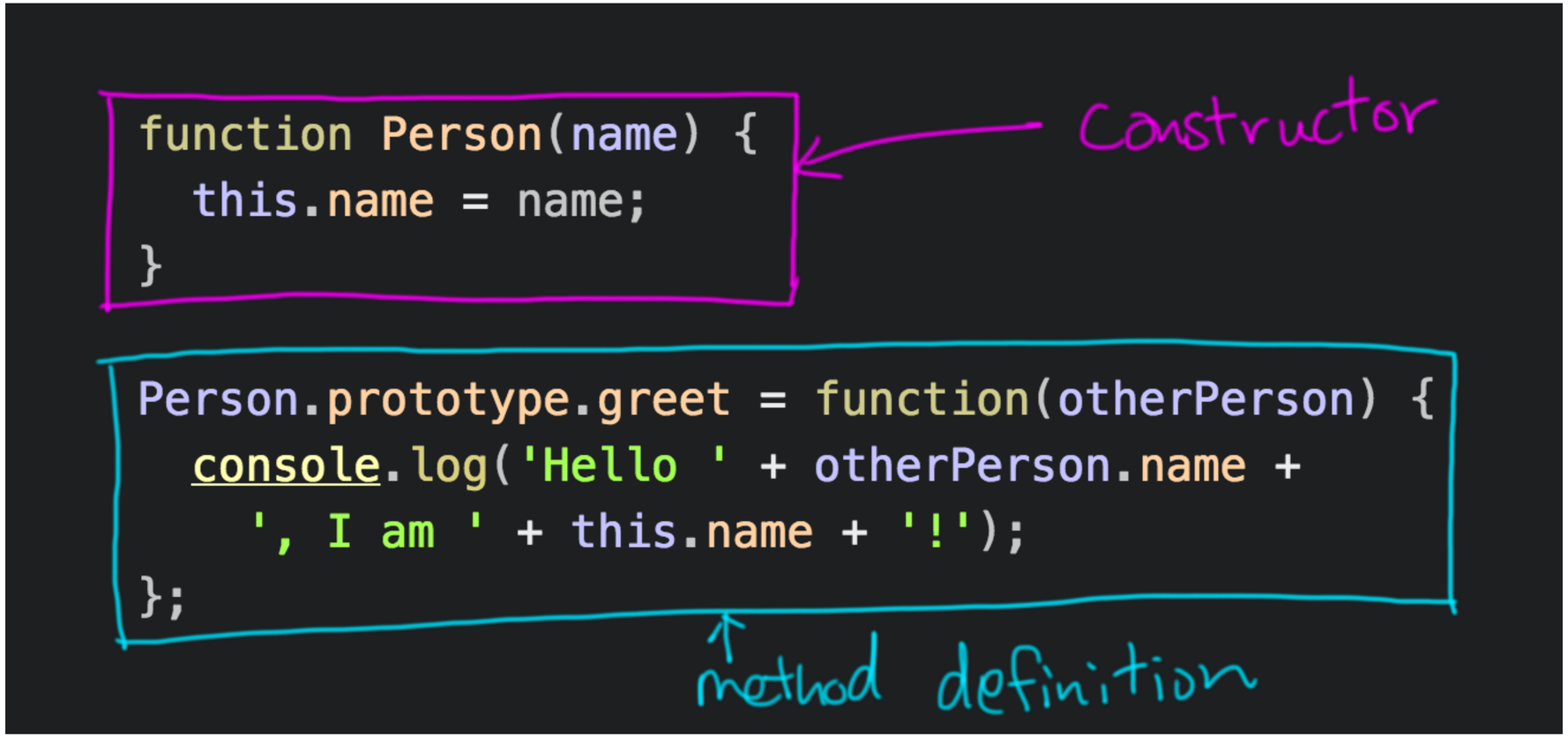
Python: Make a class

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

    def greet(self, other_person):
        print 'Hello %s, I am %s!' % (other_person.name, self.name)
```

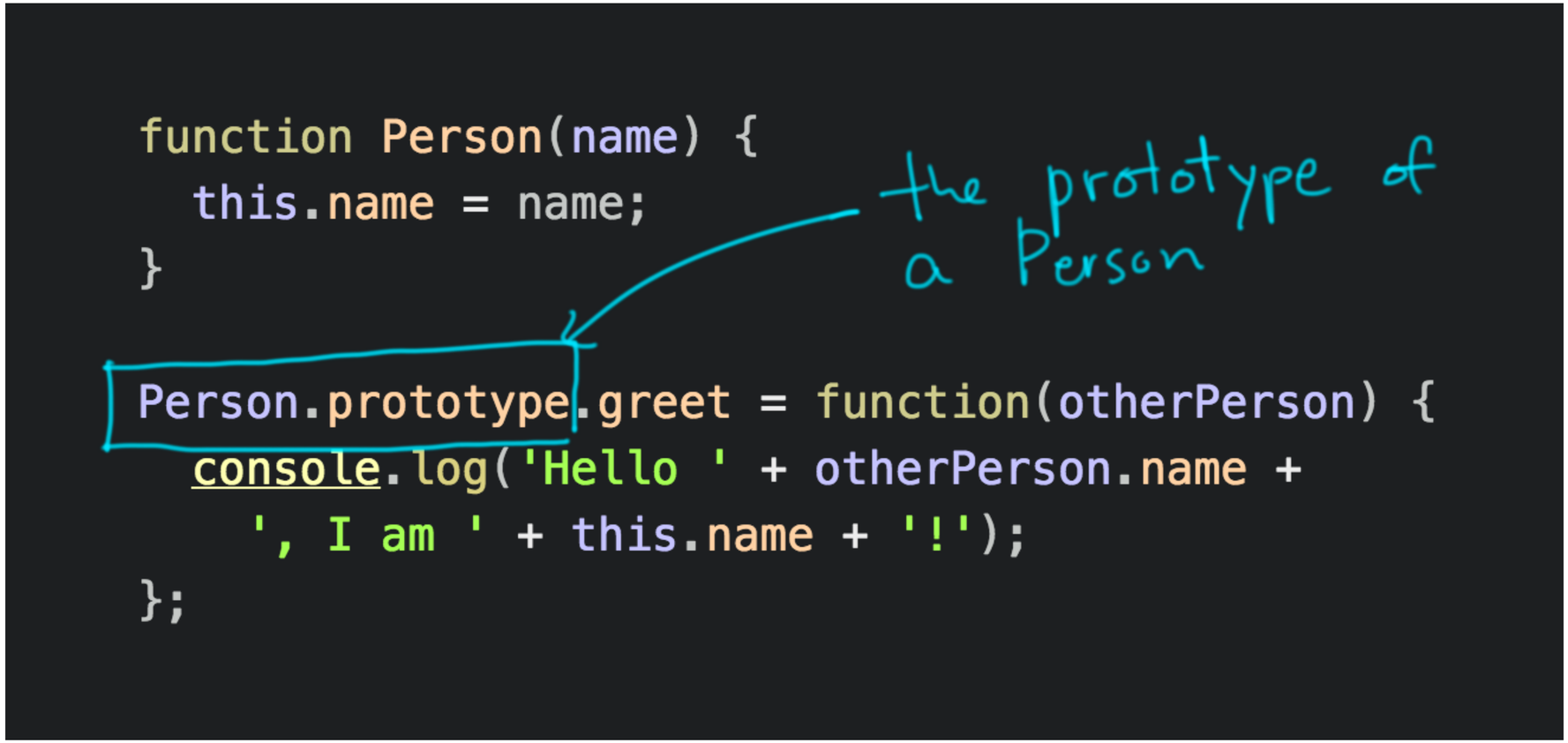
JS: Make a constructor

```
function Person(name) {
  this.name = name;
}

Person.prototype.greet = function(otherPerson) {
  console.log('Hello ' + otherPerson.name +
    ', I am ' + this.name + '!');
};
```

JS: Make a constructor



JS: Make a constructor



Side by Side

Python

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

    def greet(self, other_person):
        print 'Hello %s, I am %s!' % (other_person.name, self.name)
```

JavaScript

```javascript
function Person(name) {
  this.name = name;
}

Person.prototype.greet = function(otherPerson) {
  console.log('Hello ' + otherPerson.name + ', I am ' + this.name + '!');
};
```

Making an object

```
janice = Person('Janice')
```

```
var janice = new Person('Janice');
```

Making an object

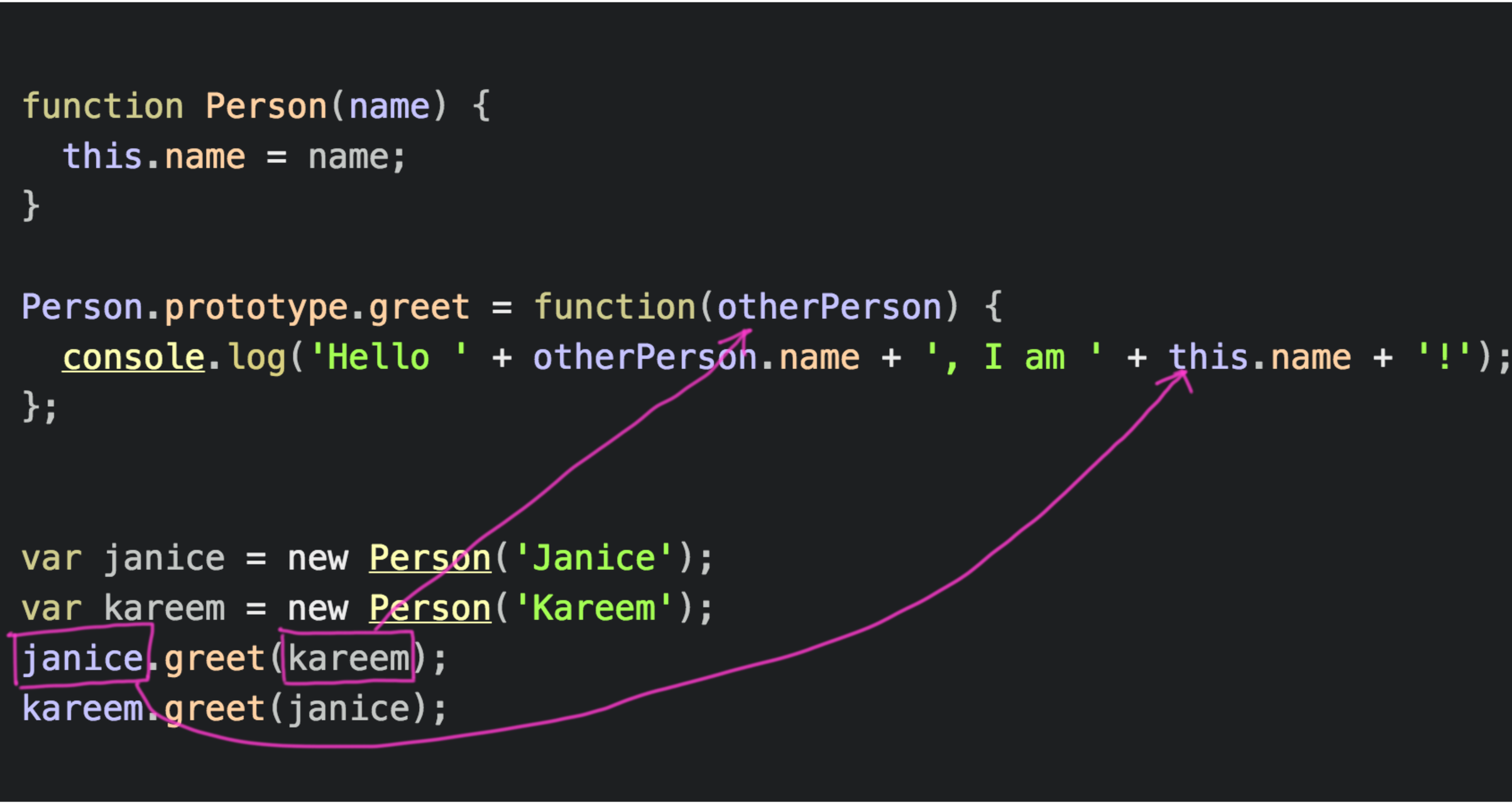
```
janice = Person('Janice')
```

```
var janice = new Person('Janice');
```

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

    def greet(self, other_person):
        print 'Hello %s, I am %s!' % (other_person.name, self.name)

janice = Person('Janice')
kareem = Person('Kareem')
janice.greet(kareem)
kareem.greet(janice)
```

```
function Person(name) {
  this.name = name;
}

Person.prototype.greet = function(otherPerson) {
  console.log('Hello ' + otherPerson.name + ', I am ' + this.name + '!');
};


var janice = new Person('Janice');
var kareem = new Person('Kareem');
janice.greet(kareem);
kareem.greet(janice);
```


object being constructed
function Person(name) {
this.name = name;
}
Person.prototype.greet = function(otherPerson) {
console.log('Hello ' + otherPerson.name + ', I am ' + this.name + '!');
};
var janice = new Person('Janice');
var kareem = new Person('Kareem');
janice.greet(kareem);
kareem.greet(janice);


function Person(name) {
this.name = name;
}
Person.prototype.greet = function(otherPerson) {
console.log('Hello ' + otherPerson.name + ', I am ' + this.name + '!');
};
var janice = new Person('Janice');
var kareem = new Person('Kareem');
janice.greet(kareem);
kareem.greet(janice);

Object Literals vs Constructor-based objects

Object Literals

```
var contact = {
  firstName: 'Janice',
  lastName: 'LaGrange',
  email: 'janicel@yahoo.com',
  phone: '485-2394-4934'
};

console.log('First name: ' + contact.firstName);
console.log('Last name: ' + contact.lastName);
console.log('Email: ' + contact.email);
console.log('Phone: ' + contact.phone);
```

Constructor-based

```
function Contact(firstName, lastName, email, phone) {
  this.firstName = firstName;
  this.lastName = lastName;
  this.email = email;
  this.phone = phone;
}

var contact = new Contact(
    'Janice', 'LaGrange', 'janice1@yahoo.com', '485-2394-4934');

console.log('First name: ' + contact.firstName);
console.log('Last name: ' + contact.lastName);
console.log('Email: ' + contact.email);
console.log('Phone: ' + contact.phone);
```