Singletons: metaclass example

```
class SingletonMetaClass(type):
    _instances = {}

    def __call__(cls, *args, **kwargs):
        if cls not in cls._instances:  # setup class instance
            cls._instances[cls] = super(
                SingletonMetaClass, cls).__call__(*args, **kwargs)
        return cls._instances[cls]

class Settings(metaclass=SingletonMetaClass):
    def __init__(self, host=None, port=None):
        print('setting up')
        self.host = host or 'localhost'
        self.port = port or 1234

alpha = Settings(host='google.com', port=5432)  # prints "setting up"
print(alpha.host, alpha.port)  # prints "google.com, 5432"
beta = Settings(host='mars.com', port=1)  # ignores input, no print
print(beta.host, beta.port)  # prints "google.com, 5432"
beta.host = 'yahoo.com'
print(alpha.host, alpha.port)  # prints "yahoo.com, 5432"
```

Semi-singletons

- Singletons, but for a particular set of inputs
- Useful for parsing into streams, distributed connections
- Uses about the same meta-class trick, but looks at input

```
class SemiSingletonMetaClass(type):
    _instances = {}

    def __call__(cls, name, *arg, **kwargs):
        if not (cls, name) in cls._instances:
            cls._instances[(cls, name)] = super(
                SingletonMetaClass, cls).__call__(*args, **kwargs)
        return cls._instances[(cls, name)]
```

Look Here