OOP & paradigms

Python Reference · Vocabulary in context

Word	Example
class /klæs/	A **class** is a plan for making objects.
object /ˈɒbdʒekt/	Each car in the game is an **object**.
instance /ˈɪnstəns/	Each dog object is an **instance** of the Dog class. Dog
constructor /kənˈstrʌktə/	The **constructor** ___init___ runs when a new object is made. ___init___
attribute /ˈætrɪbjuːt/	Name and age are each an **attribute** of a student.
method /ˈmeθəd/	upper() is a string **method**: “hi”.upper() gives “HI”. upper() ”hi”.upper() “HI”。
inheritance /ɪnˈherɪtəns/	With **inheritance**, a Bus class gets the methods of the Vehicle class. Bus Vehicle
subclass /ˈsʌbklæs/	Bus is a **subclass** of Vehicle. Bus Vehicle
superclass /ˈsuːpəklæs/	Vehicle is the **superclass** of Bus. Vehicle Bus
override /ˌəʊvəˈraɪd/	A child class can **override** a method to change what it does.
encapsulation /ɪnˌkæpsjʊˈleɪʃn/	**Encapsulation** keeps an object’s data private.
private /ˈpraɪvət/	A **private** attribute should only be used inside its own class.
polymorphism /ˈpɒlɪmɔːfɪzəm/	**Polymorphism** lets one method name work in different classes.
paradigm /ˈpærədaɪm/	Object-oriented programming is one **paradigm**.

Word	Example
procedural /prəˈsiːdʒərəl/	A **procedural** program is built from functions that run in order.
object-oriented /ˈɒbdʒekt ˈɔːrɪəntɪd/	An **object-oriented** program is built from objects with data and methods.
declarative /dɪˈklærətɪv/	SQL is **declarative**: you say what you want, not how to get it. SQL