Further Programming

A-Level Computer Science · Vocabulary in context

Word	Example
programming paradigm /ˈprəʊgræmɪŋ ˈpærədaɪm/	Object-oriented programming is a **programming paradigm**.
facts /fækts/	In Prolog, **facts** such as parent(ann, bob) state what is true. Prolog parent(ann, bob)
rule /ruːl/	A Prolog **rule** finds new facts from old ones. Prolog
low-level /ləʊ ˈlevl/	Assembly language is a **low-level** language.
registers /ˈredʒɪstəz/	The processor has a few fast **registers**.
memory addresses /ˈmeməri əˈdresɪz/	Assembly code uses **memory addresses** directly.
objects /ˈɒbdʒekts/	A program can make many **objects** from one class.
attributes /ˈætrɪbjuːts/	A car's **attributes** include its colour and speed.
methods /ˈmeθədz/	A car's **methods** include Drive() and Stop(). Drive() Stop()
machine code /məˈʃiːn kəʊd/	The processor runs **machine code**: only 0s and 1s. 0 1
assembly language /əˈsemblɪ ˈlæŋgwɪdʒ/	**Assembly language** uses short words such as ADD and LDM. ADD LDM
addressing modes /əˈdresɪŋ məʊdz/	Direct and indirect are two **addressing modes**.
array /əˈreɪ/	An **array** stores many values under one name.

exercises.lol exercises.com.cn

Word	Example
imperative programming /ɪmˈperətɪv ˈprəʊgræmɪŋ/	In **imperative programming**, you tell the computer each step.
Variables /ˈveərɪəblz/	**Variables** store the data in a program.
object-oriented programming /ˈɒbdʒekt ˈɔːrɪəntɪd ˈprəʊgræmɪŋ/	**Object-oriented programming** builds programs from objects.
instances /ˈɪnstənsɪz/	Each car is one of the **instances** of the Car class. Car
classes /ˈklæsɪz/	Car and Bus are **classes**. Car Bus
encapsulation /ɪnˌkæpsjʊˈleɪʃn/	**Encapsulation** keeps an object's data private.
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
overriding /ˌəʊvəˈraɪdɪŋ/	**Overriding** gives a subclass its own version of a method.
polymorphism /ˈpɒlɪmɔːfɪzəm/	**Polymorphism** lets one method name work in different classes.
abstraction /əbˈstrækʃn/	A map uses **abstraction**: it leaves out small details.
constructor /kənˈstrʌktə/	The **constructor** sets up a new object.
aggregation /ˌægrɪˈgeɪʃn/	**Aggregation**: a class has students, but the students exist without it.

Word	Example
containment /kən'teɪnmənt/	**Containment**: a house contains rooms.
declarative programming /dɪ'klærətɪv 'prəʊgræmɪŋ/	In **declarative programming**, you say what you want, not how.
functional programming /'fʌŋkʃənl 'prəʊgræmɪŋ/	**Functional programming** builds programs from functions.
pure functions /pjʊə 'fʌŋkʃnz/	**Pure functions** always give the same output for the same input.
side effects /saɪd ɪ'fekts/	A function with **side effects** changes something outside it.
logic programming /'lɒdʒɪk 'prəʊgræmɪŋ/	Prolog is used for **logic programming**. Prolog
SQL /ˌes kju: 'el/	**SQL** is a language for asking a database questions.
file /faɪl/	Save the scores in a **file**.
random file /'rændəm faɪl/	In a **random file**, a hash decides where each record goes.
hashing /'hæʃɪŋ/	**Hashing** turns a key into a number.
exception /ek'sepʃn/	Dividing by zero raises an **exception**.
exception handling /ek'sepʃn 'hændlɪŋ/	**Exception handling** stops the program from crashing.
raise /reɪz/	The program will **raise** an error if the file is missing.

exercises.lol exercises.com.cn

call stack /kɔ:l stæk/	Each function call is added to the **call stack**.