Creative Development

AP Computer Science Principles · Vocabulary in context

Word	Example
collaborative /kə'læbrətɪv/	A **collaborative** project uses ideas from the whole team.
perspectives /pə'spektɪvz/	People with different **perspectives** spot different problems.
consensus building	**Consensus building** helps the team agree on one design.
Pair programming /peə 'prəʊgræmɪŋ/	In **pair programming**, two people share one computer.
bug /bʌg/	The team found a **bug** that made the score go below zero.
testing /'testɪŋ/	Use **testing** to compare actual and expected answers.
computing innovation /kəm'pju:tɪŋ ˌɪnə'veɪʃn/	A **computing innovation** is anything new that needs a program to work.
bias /'baɪəs/	A program trained on unfair data can show **bias**.
user /'ju:zə/	The **user** is the person who uses the program.
conflict resolution	Good teams use **conflict resolution** to settle disagreements.
negotiation /nɪˌgəʊʃi'eɪʃn/	**Negotiation** helps the team find a plan everyone accepts.
purpose /'pɜ:pəs/	Ask about the **purpose** of the text: what must I do?

exercises.lol exercises.com.cn

Word	Example
input /'ɪnpʊt/	input() reads **input** that the user types. input()
output /'aʊtpʊt/	The **output** of print(2 + 3) is 5. print(2 + 3) 5
program /'prəʊgræm/	A **program** is a list of instructions for the computer.
software /'sɒftweə/	**Software** is the programs on a computer.
code segment	A **code segment** is one part of a program.
behavior /bɪ'heɪvjə/	The limit describes the **behavior** of f(x) near x = 2. f(x) x = 2
tactile	A phone's vibration is **tactile** output.
iterative /'ɪtərətɪv/	The **iterative** process repeats: design, test, improve.
diagram	She drew a **diagram** of the screens before writing any code.
implement	After the design is agreed, the team can **implement** it in code.
decomposition /ˌdi:kɒmpə'zɪʃn/	**Decomposition** breaks a big problem into small parts.
Comments /'kɒments/	**Comments** explain the code to people but do not run.
incremental /ˌɪŋkrɪ'mentl/	In **incremental** development, you add and test one feature at a time.
prototype /'prəʊtəʊtaɪp/	A **prototype** is an early version of the program.
surveys /'sɜ:veɪz/	We used **surveys** to ask users what they wanted.

Word	Example
interview /ˈɪntəvjuː/	The radio **interview** is with a young chef.
observation /ɒbzəˈveɪʃn/	In an **observation**, the researcher watches behaviour.
user testing	**User testing** means watching real people try the program.
program requirements	The **program requirements** say the app must work offline.
user interface /ˈjuːzə ˈɪntəfeɪs/	The **user interface** is the screen you use to control the program.
specification /ˌspesɪfɪˈkeɪʃn/	The **specification** lists what the program must do.
storyboard /ˈstɔːribɔːd/	A **storyboard** sketches each screen in order.
event /ɪˈvent/	A mouse click is an **event**.
flow of execution	A function call changes the **flow of execution**.
event handler /ɪˈvent ˈhændlə/	The **event handler** runs when the user clicks the button.
event-driven programming	In **event-driven programming**, code runs when the user clicks or types.
program documentation	**Program documentation** explains what the code does.
debugging /ˈdiːbʌgɪŋ/	**Debugging** means finding and fixing mistakes in a program.
syntax error /ˈsɪntæks ˈerə/	A missing bracket is a **syntax error**.

exercises.lol exercises.com.cn

Word	Example
runtime error /ˈrʌntaɪm ˈerə/	Dividing by zero causes a **runtime error**.
logic error /ˈlɒdʒɪk ˈerə/	A **logic error** gives the wrong answer but does not stop the program.
overflow error /ˌəʊvəˈfləʊ ˈerə/	An **overflow error** happens when a number is too big for the computer.
test case	A **test case** gives an input and the output you expect.
print statement	Add a **print statement** inside the loop to see the total change.
hand-tracing	**Hand-tracing** the loop showed the error on the third pass.
debugger /ˈdiːbʌgə/	The **debugger** helps you find mistakes.
visualization /ˌvɪʒuːəlaɪˈzeɪʃn/	A chart is a **visualization** that makes patterns easy to see.
order of operations /ˈɔːdə ɒv ˌɒpəˈreɪʃnz/	The **order of operations** says multiply before you add.
feedback /ˈfiːdbæk/	**Feedback** lets the system check the result and adjust.