Thinking like a programmer: big ideas

Idea	Thinking like a programmer—details	Section
Setting up	Don't start to code immediately. You'll feel boxed into one path that may or may not even be appropriate for the problem at hand.	19.1.2
Setting up	If you find yourself writing convoluted logic to achieve a simple task or repeating yourself several times, take a step back and use a piece of paper to draw out what you want to achieve.	17.2
Setting up	When thinking about how to break down your problem, choose tasks and write tasks in such a way that they can be reusable.	29.1
Setting up	Before beginning to code, think about each data type you've learned about and decide whether it's an appropriate one to use. When more than one may work, pick the simplest one.	29.3
Setting up	When choosing data attributes to represent an object type, you can (1) Write out the data types you know and ask whether each would be appropriate to use. (2) Notice whether the behaviors you want can be represented by one or more data structures you already know.	32.1.1
Setting up	Think about expressions that contain decisions as Boolean expressions (which evaluate to true or false) as opposed to questions (which have yes or no answers).	13.1.1
Setting up	Computers do only what they're told. When writing code, the computer will execute everything you write according to the rules of the programming language.	16.1.2
Readability	A programmer writes readable code, both for others to be able to read as well as for themselves to look back on later. Use descriptive variables to store complex computations.	14.2.2
Readability	Don't use variable names that are very long. They make your code unreadable.	4.2.3
Readability	Create variables to store values that you're going to reuse many times in your code.	18.3.1
Readability	Write code that's as simple and short as possible while being easy to understand.	18.2
Abstraction	You hide implementation details from people by using your class. Users don't need to know the implementation to use the class.	32.1
Abstraction	Before running a function on large input files, try it on a smaller test file.	29.3
Modularity	Divide your code into smaller pieces and test each one separately. After you know that each separate piece works as expected, you can put the pieces together to create your final program.	19.1.2
Modularity	Functions should be self-contained pieces of code that you need to debug	20.2

Idea	Thinking like a programmer—details	Section
Debugging	Using descriptive names for methods is useful in providing quick, at-a-glance information when tests fail.	36.2
Debugging	While debugging, trace through a program. Go line by line, draw the scope you're in, and write down any variables and their values currently in the scope.	22.2.2
Documentation	When faced with writing a line of code, browse the Python documentation to see what functions you can use before writing your own.	23.1.3
Documentation	A programmer tries to make life easier for other programmers. Document your classes and methods, and whenever possible, implement special methods.	33.3

Recommended / Playlists / History / Topics / Settings / Get the App / Sign Out

PREV
Appendix C. Interesting Python libraries

NEXT
Index