|epcc|

Code Review

B087928

October 15, 2015

1 Introduction

Good coding practise extends far further than having a code without any bugs in it. Often we work on a project with other people and so it is important for them to be able to read and understand our code. This means that it is important to have well structured and designed code. The code must also be well documented to illustrate what more complicated parts are doing. This documentation is redundent if used on obvious pieces of code unless we explain why we are doing it. In this report we examine a simple piece of code which has no known bugs, and discuss what parts of the code are bad practise and how we'd improve on them.

2 Code Review

2.1 How easy it is to read

The main function in the code takes our input value as an argument and prints all the powers of two that add to make that value. Functions should usually have names that suggest their function or where we should use them. Therefore the name "p2" is useless. It would be far more useful to name this function "`cal_binary_array`" or simply "`powers`".

Similarly, the name of the program should idicate to the programmer what to use it for. "`program.py`" gives us no information other than it is a python script and so will be indistinguishable from other python scripts.

We can see that "ps" is declared as an array at the top of the code. This, however, is not very clear from the rest of the code. This would be confusing if we have longer code with several arrays, each with different functions. It would be more useful to call this "`powers_array`".

2.2 How easy it is to understand

One of the biggest obstacles in understanding a piece of code is misinterpreting the meaning of a variable or action. Naming a variable "Sum" suggests that it is incremented in a loop somewhere in the code. Whereas, here it is simply the input value which, although is a sum of the final output of the program, misleads the reader and will make it take longer to understand. It is also a source of bad code design as sum is math function. For these reasons 'value' is a better choice of name.

`$ if(sum %2)` : Executes the structured block if the argument is one and doesn't execute if argument is zero. This is exactly what the code is supposed to do. However, a glance at this section of the code does not make it at all clear. It would be much more understandable if the syntax was changed to: `if(sum%2 == 1)` or `if(sum %2 != 0)`.

"x" is used in the code and then again later as a loop index. This may mislead the reader as to what the loop is doing. It would be more appropriate to use a new variable, say, "i" as the loop index.

2.3 How well the code is structured

It is important that we structure code well: it is cumbersome and inefficient to have any section of the code that is redundant.

The function "`p2`" returns the array "`ps`". This is unnecessary as the function call does not set a vaule to the return. The function does everything required of it by printing the array ps. This return statement should be excluded for better designed code.

The if statement `if __name__=='__main__':` is unnecessary since we have one simple script and can be excluded.

2.4 How well documented

We shall now examine the comments of the code.

`x=x*2 #Multiply by 2. Is this a bug?` This is a useless comment as the code clearly multiplies the variable "x" by 2. What would be more useful here is an explanation of why we want to multiply "x" by 2. Therefore this comment can be replaced by `#To a higher power of 2`. Is this a bug is also a useless comment. It alerts the reader that we are unsure, or would like to double check that piece of code, but doesn't tell us why. A more appropriate and helpful comment would be `#Need to re-examine if we can double x in this way`.

`sum=sum >> 1 #Do a shift`. What is a shift and why are we doing it? This action drops the rightmost binary digit and shifts each binary digit to the right such that the value at 2^2 is now at 2^1 and so on. If the reader didn't know what is meant by a shift then they won't understand this comment, if they do understand this comment then this is not useful. Instead the comment should explain why we'd like to do a shift.

`#Print the powers`. This would read better as `Print the powers of two that sum to the input value`.

3 Conclusions

It is wrong to think that if code works correctly and has no bugs that it has been programmed well. We want others to be able to easily read and understand our code. Good documentation, structure and readability is also very important if we want to re-examine or change our code after a period of time. We may forget what sections of our code does or is used for so well programmed code largely speeds up our re familiarisation.