Tips to become a better programmer and ace a

In [ ]:
```
# Let's start with a sample problem: compute the product of all elements in list L
# (L[0]*L[1]*L[2]*...)

L = [1, 2, 3]

prod = 1
for elem in L:
    prod *= elem
print(prod)
```

Best coding practices:

1. Always use length of size to loop through arrays and lists (avoid using magic numbers)
2. Use understandable variable names (not too long but also not one letter usually)
3. Use minimal syntax
4. Always check for edge cases (inputs that are different from typical inputs)
5. Make your code as efficient as possible
6. BONUS: Comment your code and use white space appropriately

In [1]:
```
# Let's look at efficiency - given some list of numbers L, compute the average of the numbers
# in the list. Then append a number to the list and recompute the average.

L = [0]*100000000
#L.append(new_number)     <- appending a number to the end of the list

s = 0
for elem in L:
    s += elem
avg = s / len(L)
print(avg)

#append a number and compute the average
new_num = 10
L.append(new_num)

s = 0
for elem in L:
    s += elem
avg = s / len(L)
print(avg)
```

```
0.0
9.9999999e-08
```

In [2]:
```
L = [0]*100000000
#L.append(new_number)     <- appending a number to the end of the list

s = 0
for elem in L:
    s += elem
avg = s / len(L)
print(avg)

#append a number and compute the average
new_num = 10
L.append(new_num)

s += new_num
avg = s/len(L)
print(avg)

# this code is more efficient because it does half as much the computation

#6.76 seconds
```

```
0.0
9.9999999e-08
```

Cracking the coding interview

Now that you know all of the best technical practices for a coding interview, let's talk about behavioural practices and what to expect.

In an interview, you should know the following topics

1. Arrays
2. strings
3. Dynamic programming*
4. Graph algorithms (should know stacks, queues, bredth first search, and depth first search)
5. Recursion
6. Hashing and maping (in python, you should know about dictionaries and sets)
7. Searching and sorting
8. Trees

What companies are looking for (during the interview)

1. Communication. Are you asking for clarification or just diving into the code?
2. Problem solving. Are you providing the most efficient solution?
3. Coding. Can you convert your ideas into bug-free code?
4. Verfication: Are you testing on edge case and simple examples?

Usually, you will be asked to solve 2 problems in 40 minutes.

To prep for this type of interview

1. Take a data structure or algorithms class like CS 2110
2. Practice with easy and medium LeetCode problems

During the interview

1. You get to choose which language you want to code in.
2. Make sure you fully understand the question being asked. If you can't do it by hand, you can't code it up.
3. Always check edge cases
4. Talk!

In [7]:
```
# Problem to try: Given an array nums of size n, print the majority element. The majority element
# is the element that appears more than n/2 times. Assume there is a majority element in nums.

nums = [1, 2, 2, 10, 0, 2, 2]
n = len(nums)

dict = {}
for elem in nums:
    if elem not in dict:
        dict[elem] = 1
    else:
        dict[elem] += 1

for key in dict:
    if dict[key] > n/2:
        print(key)
```

```
2
```

In [8]:
```
n = [1, 2, 2, 10, 0, 2, 2]

store = (len(n)/2)
element = "none"
for num in n:
    if n.count(num) > store:
        store = n.count(num)
        element = num

print(element)
```

```
2
```

In [ ]:
```
nums = [1, 2, 2, 10, 0, 2, 2]
n = len(nums)

for elem in nums:
    e = nums.count(elem)
    if e > n/2:
        maj_elem = elem
print (maj_elem)
```

In [ ]:
```
# Given a string s consisting of words and spaces, print the length of the last word in the string.
# A word is a substring consisting of non-space characters only. No punctuation and no numbers

# some examples
s = "a great big world"
s = "a"
s = "      "
s= "one"
s = "a great big world                    "

#two ways to loop from the end of an array
for i in range(len(s)):
    print(s[-1-i])

for i in range(len(s)):
```

In [16]:
```
s = ""

s = s.strip()
space = s.rfind(" ")
last = s[space+1:]
print(len(last))
```

```
0
```

In [ ]:
```
# problem to try at home: given a natural number int, write a function that outputs true if
# the number is a palindrome or false if the number is not a palidrome.
```