Solutions to the provided "broken" code

```
friends = ['john', 'pat', 'gary', 'michael']
for i, name in enumerate(friends):
    # friends is a list of strings, so we could simply change the second
    # formatting parameter to %s
    # However, we modify the statement to use .format, as it is strongly
    # preferred and % is being deprecated.
    print "name {0} is {1}" .format(i, name)


# How many friends contain the letter 'a' ?
count_a = 0
for name in friends:
    #correctly represent the character a as a one-character string
    if 'a' in name:
        #correctly increment count_a in Python as opposed to C++
        count_a+=1

# Explicitly cast count_a to float to avoid integer division rounding
# (in this case, down)
# Casting len(friends) as a float will also work.
# Also changed the print statement to use .format, as both it is strongly
# preferred and % is being deprecated.
print "{} percent of the names contain an 'a'".format(( float(count_a) / len(friends) ))


# Say hi to all friends
# Corrected having the optionoal parameter first
# Corrected appropriate string concatenation
def print_hi(name,greeting='hello'):
    print greeting +' ' +name

map(print_hi, friends)
```

```
# Print sorted names ou
# Corrected to use sorted method
print sorted(friends)


# Assuming this was meant to be a comment block, and not a docstring,
# corrected to comment each  line. Python has no official block coding
# mechanism, (though docstring notation is often used that way, as if
# it's not in the beginning then it's ignored), so a # needs to be
# placed before each line. This is simple, as almost every IDE has a
# key combination for commenting a block automatically.
#
#    Calculate the factorial N! = N * (N-1) * (N-2) * ...
#
def factorial(x):
    """
    Calculate factorial of number
    :param N: Number to use
    :return: x!
    """
    # Though we are aware of the Gamma function and extending factorial
    # to non-integers, we assume that is not the purpose here. Similarly,
    # we do not check the size of x to determine if we should use the
    # Stirling formula instead of crashing the machine.
    # Thus, we prevent unwanted behavior by requring x be a positive integer
    try:
        assert x == int(x) and x>=0
    except AssertionError:
        print "NaN : input must be a positive integer"
        return None

    # Corrected function to appropriately include 0
    if x==0 or x==1: return 1
    #corrected the recursive relationship by prepending x* to the recursive call.
    return x*factorial(x-1)
```