```
""" ants1.py
    classic counting song
    demonstrates use of multi-line strings
    first version
    """


def chorus():
    """ prints chorus """
    print """
    ...and they all go marching
    down-
    to the ground-
    to get out-
    of the rain.
    Boom boom boom boom boom boom boom
    """

def verse1():
    """ prints first verse """
    print """
    The ants go marching 1 by 1 hurrah, hurrah!
    The ants go marching 1 by 1 hurrah, hurrah!
    The ants go marching 1 by 1,
    The little one stops to suck his thumb
    """

def verse2():
    """ prints second verse """

    print """
    The ants go marching 2 by 2 hurrah, hurrah!
    The ants go marching 2 by 2 hurrah, hurrah!
    The ants go marching 2 by 2,
    The little one stops to tie his shoe
    """

verse1()
chorus()
verse2()
chorus()
```

```
""" ants2.py
    classic counting song
    use parameters and return statements
    """


def chorus():
    output = """
    ...and they all go marching
    down-
    to the ground-
    to get out-
    of the rain.
    Boom boom boom boom boom boom boom
    """
    return output

def verse(verseNum):
    if verseNum == 1:
        distraction = "suck his thumb"
    elif verseNum == 2:
        distraction = "tie his shoe"
    else:
        distraction = "I have no idea"


    output = """
    The ants go marching %(verseNum)d by %(verseNum)d hurrah, hurrah!
    The ants go marching %(verseNum)d by %(verseNum)d hurrah, hurrah!
    The ants go marching %(verseNum)d by %(verseNum)d,
    The little one stops to %(distraction)s
    """ % vars()
    return output

print verse(1)
print chorus()
print verse(2)
print chorus()
```

```
""" ants3.py
    classic counting song
    further improvement with a list
"""

distraction = [
      "",
      "suck his thumb",
      "tie his shoe",
      "climb a tree"
      ]


def chorus():
    output = """
    ...and they all go marching
    down-
    to the ground-
    to get out-
    of the rain.
    Boom boom boom boom boom boom boom
    """
    return output

def verse(verseNum):

    problem = distraction[verseNum]
    output = """
    The ants go marching %(verseNum)d by %(verseNum)d hurrah, hurrah!
    The ants go marching %(verseNum)d by %(verseNum)d hurrah, hurrah!
    The ants go marching %(verseNum)d by %(verseNum)d,
    The little one stops to %(problem)s
    """ % vars()
    return output

for verseNum in range(1, len(distraction)):
    print verse(verseNum)
    print chorus()
```

```
""" scope.py
    illustrates scope and functions
    """

varOutside = "I was created outside the function"
print "outside the function, varOutside is: %s" % varOutside

def theFunction():
    varInside = "I was made inside the function"

    print "inside the function, varOutside is: %s" % varOutside
    print "inside the function, varInside is: %s" % varInside

theFunction()

print "back outside the function, varOutside is: %s" % varOutside
# if I uncomment the next line, the program will crash
# print "back outside the function, varInside is: %s" % varInside
```