```cpp
//  file: area.cpp
//
//  This program calculates the area of a circle, given the radius.
//
//  Programmer:  Dick Furnstahl  furnstahl.1@osu.edu
//
//  Revision history:
//     02-Jan-2004  original version, for 780.20 Computational Physics
//     01-Jan-2010  updates to "To do" wishlist
//
//  Notes:
//   * compile with: "g++ -o area area.cpp"
//
//  To do:
//   1. output the answer with higher precision (more digits)
//   2. use the "predefined" value of pi or atan
//   3. define an inline square function
//   4. split the calculation off into a function (subroutine)
//   5. output to a file (and/or input from a file)
//   6. rewrite using a Circle class
//
//*************************************************************************//

// include files
#include <iostream>       // this has the cout, cin definitions
using namespace std;      // if omitted, then need std::cout, std::cin

//*************************************************************************//

const double pi = 3.1415926535897932385;   // define pi as a constant

int
main ()
{
  double radius;     // every variable is declared as int or double or ...

  cout << "Enter the radius of a circle: ";              // ask for radius
  cin >> radius;

  double area = pi * radius * radius;   // area formula

  cout << "radius = " << radius << ", area = " << area;

  return 0;                       // "0" for successful completion
}
//*************************************************************************//
```

```
SHELL=/bin/sh
# Brief notes on makefiles:  Comments start with #.  $(COMMAND) means
#  replace with the value of COMMAND assigned with an "=".

# This file contains a set of rules used by the "make" command.
#   This makefile $(MAKEFILE) tells "make" how the executable $(COMMAND)
#   should be generated from the source files $(SRCS) and the header files
#   $(HDRS) via the object files $(OBJS); type the command:
#      "make -f make_program"
#   where make_program should be replaced by the name of the makefile.
#
# To remove the OBJS files; type the command:
#      "make -f make_program clean"
#
# To create a zip archive with name $(COMMAND).zip containing this
#   makefile and the SRCS and HDRS files, type the command:
#      "make -f make_program zip"

# The name of this makefile goes here
MAKEFILE= make_area

# The command you type to run the program (executable name)
COMMAND=  area

# Here are the C++ (or whatever) source files to be compiled, with \'s as
#  continuation lines.  If you get a "missing separator" error pointing
#  to a line here, make sure that each \ has NO spaces following it.
SRCS= \
area.cpp

# Header files (if any) here
HDRS= \

#########################################################################
# Commands and options for compiling
#########################################################################
OBJS= $(addsuffix .o, $(basename $(SRCS)))

CC= g++
CFLAGS=  -g -O3
WARNFLAGS= -Werror -Wall -W -Wshadow -fno-common
MOREFLAGS= -ansi -pedantic -Wpointer-arith -Wcast-qual -Wcast-align \
           -Wwrite-strings -fshort-enums
LDFLAGS= -lgsl -lgslcblas

#########################################################################
# Instructions to compile and link -- allow for different dependencies
#########################################################################

$(COMMAND): $(OBJS) $(HDRS) $(MAKEFILE)
        $(CC) -o $(COMMAND) $(OBJS) $(LDFLAGS) $(LIBS)

area.o : area.cpp $(MAKEFILE)
        $(CC) $(CFLAGS) $(WARNFLAGS)  -c area.cpp -o area.o

#########################################################################
# Additional tasks
#########################################################################

clean:
        rm -f $(OBJS)

zip:
        zip -r $(COMMAND).zip $(MAKEFILE) $(SRCS) $(HDRS)

#########################################################################
# End of makefile
#########################################################################
```

```python
#  file: area0.py
#
#  This program calculates the area of a circle, given the radius.
#
#  Programmer:  Dick Furnstahl  furnstahl.1@osu.edu
#
#  Revision history:
#      26-Dec-2008  original version, translated from area.cpp
#
#  Notes:
#   * run program using "python area.py"
#   * conversion from .cpp to .py:
#         * // --> # for comments on a single line
#         * drop the semicolons
#         * no variable declarations like int or double
#         * radius**2 instead of radius*radius
#         * different functions for input and output
#
#  To do:
#   * output the answer with higher precision (more digits)
#   * split the calculation into a function (def)
#   * output to a file (and/or input from a file)
#
#***********************************************************************#

pi = 3.141592653589793     # put in \pi by hand

answer = raw_input('Enter the radius of a circle: ')  # answer is a string
radius = float(answer)   # convert to floating point number

area = pi * radius**2     # area formula; x**n is x to the n'th power

# simple printing (illustrates that either type of quotes can be used)
print 'radius = ', radius, ', area = ', area
print "radius = ", radius, ", area = ", area

# That's all, folks!
```

```python
"""
file: area1.py

This program calculates the area of a circle, given the radius.

Programmer:  Dick Furnstahl  furnstahl.1@osu.edu

Revision history:
   26-Dec-2008  original version, modified from area0.py

Notes:
 * run program using "python area.py"
 * conversion from area0.py:
    * use three quotes for multiline comments instead of #
    * use the value of pi defined in the math module
    * do conversion from raw_input to float in one line

To do:
 * output the answer with higher precision (more digits)
 * split the calculation into a function (def)
 * output to a file (and/or input from a file)
"""

import math     # read in the definitions from the math module

# Just do it!

# convert the input to a float right away
radius = float(raw_input('Enter the radius of a circle: '))

area = math.pi * radius**2     # area formula A = pi R^2

# partially formatted print: compare %f to %e
print 'radius = %f, area = %f' % (radius, area)
print 'radius = %e, area = %e' % (radius, area)

# now some additional digits (%.nf means n digits after decimal)
print 'radius = %.4f, area = %.8f' % (radius, area)

# That's all, folks!
```