1. Ask the user for a float value then show her the value:

\[ \cos \sqrt{x^3} \]

You do NOT have to convert the number into radians.  
Do write the required import(s) that would be needed.

```python
import math

def cosSqrtOfCube( value ):
    ''' returns the cosine of the square root ot value cubed '''
    return math.cos( math.sqrt( value ** 3 ) )

def showUserCosSqrtValueCubed():
    '''
    gets a float from the user and shows
    cos( sqrt( value**3 ) ) on screen
    assumes float input
    '''
    value = float( raw_input( "Enter a value: " ) )
    result = cosSqrtOfCube( value )
    print 'calcuation result is: ' + str( result )
```

2. same as 1.

3. Write the function named `oneIsPositive` that returns `True` when any of it's three parameters is positive

```python
def oneIsPositive( first, second, third ):
    ''' returns True if any parameter is posivie, False otherwise'''
    return first > 0 or second > 0 or third > 0
```

Notice that there is NO if-else statement here – just return the bool value

```
WRONG   WRONG   WRONG   WRONG   WRONG   WRONG   WRONG   WRONG:
WRONG:  if first > 0 or second > 0 or third > 0:
WRONG:     return True
WRONG:  else:
WRONG:     return False
```

4. What is the relationship between the return type of a function and the types of the parameters?  
There is NO relationship.  
The types of the parameters are described in the docstring to let the caller know what kinds of info is needed by the function to do its job.  
The type of the returned value is what the caller needs back from the function call (it's type is also documented in the docstring)
5. How many values can a function return to its caller?
   You can answer this one in of two ways since you now know a little about lists
   As many as the caller needs back – really you are only returning one list
   return value1, value2, value3, ....

6. How does a programmer communicate with the user?
   print and raw_input
   How does a programmer communicate with the programmer who calls her functions?
   parameters for data coming into the function, return for data going back to the caller
   Do both mechanisms have to be in every function?
   No
   Is writing a value on the screen communicating with the caller of a function?
   Absolutely NOT!

7. What are the three ways a programmer writing a function def can get information?
   - 1 parameters which force the caller to provide information for each parameter
   - 2 global consts
   - 3 input. this is only to be done when the user is who knows the information while the function is running. If the caller knows that information, do NOT ask the user
   - and there is a fourth – described below

8. What is wrong with the call of function f?

   def f( nameOfAnimal ):
       ''' returns whether or not the name is in our database '''
       ...
   if( f( 3 ) ):

   The caller didn't read the docstring before calling the function – they passed in an int but the docstring clearly indicates that a string (the name of an animal) is what the function needs

9. What is the parameter passing mechanism in python?
   "pass by assignment"
   What does that mean in terms of RAM and referrals?
   The parameter name and the caller's value refer to the same place in RAM
   the GSitS (the Great Snake in the Sky (the interpreter) passes the value at the call to the function as if it had written parameterName = valuePassedAtCall:

   def f( parameterName ):
       you can think that
       the GSitS does:
       parameterName = 3
       before the rest of the body starts executing
       the 3 came from the x referring to 3 at the call

   def someOtherFunction():
       x = 3
       f( x )
10. How many copies are there of the data passed into a function? (see previous question!) Only one copy of the data BUT – if the function reassigns the parameter name, nothing happens to the caller's variable – still only one copy of each value – the parameter is made to refer to a different place in RAM when reassigned.

```python
def f( parameterName ):
    parameterName = 4
```

```python
def someOtherFunction():
    x = 3
    f( x )
in RAM:
    3
    4
```

11. What is the scope of a parameter? of a local variable?

Both are Local to the function.

12. Why does this show a misunderstanding? What is being misunderstood

```python
def puny( bird ):
    
    def func( bird2 ):
        
Scope is being misunderstood.
Why dose the programmer writing `func` think they need that `bird`?
There is no other `bird` in the local scope of `func`.
The programmer writing `func` seems to think that there will be a conflict unless they choose a different name.

13. def main must be the first def in a python program (true or false)
false
14. What is wrong with this code? The correct breakdown is stepONE, stepTWO, stepTHREE. So that must be the order of execution for the programmer defining main. Here *EACH* function somehow knows what the next step should be. Only the programmer calling these functions would know that order.

```python
def main():
    stepONE()

def stepTWO( ):
    ''' does step two '''
    # code to do step two goes here
    stepTHREE()

def stepTHREE( ):
    ''' does step three '''
    # code to do step three goes here

def stepONE():
    ''' does step one '''
    # code to do step one goes here
    stepTWO()

main()
```

In main there appears to be only ONE step to solve the problem. This ERROR is called chaining functions. stepONE has NOTHING to do with calling the next step – it's designer cannot need to know when it will be called and cannot need to know which function should be called after it:

Here is how the program should (MUST) be written:

```python
def main():
    stepONE()
    stepTWO()
    stepTHREE()

def stepTWO( ):
    ''' does step two '''
    # code to do step two goes here

def stepTHREE( ):
    ''' does step three '''
    # code to do step three goes here

def stepONE():
    ''' does step one '''
    # code to do step one goes here

main()
```
15. There is actually a fourth way a programmer can get data in a function.
   What is it?
We mostly discuss how to get information from outside the function into the function by:
   global consts
gotten from the caller through parameters
input from the user and stored locally in (local) variables

The fourth is:
   calculated directly or returned in a function call
   and stored locally in (local) variables

16. Does raw_input have a return statement in it?
   How else could it return a string to its caller? – so Yes!

17. State the four branching problem forms and the four looping problem forms.
18. State the python statements for each of the four branching problem forms and the four looping
    problem forms from Q17

branching:
   do or don't do
      ------------------if statement
   do one of two
      ------------------if-else statement
   do one of many, one must
      ------------------if-elif-elif-….else statement
   do one of many or possibly none
      ------------------if-elif-elif-….elif statement

looping:
   unknown number of times to do something but can test for end
      ------------------while statement
   known or knowable (via user input or calculation) number of times to do something
      ------------------for loopVar in range( #times):
   processing a sequence of values
      ------------------for loopVar in range( start, stop, step):
   ------------------for loopVar in [ literal-written-out-sequence-of-values ]:
   infinite loop (probably with conditional middle exit)
      ------------------while True:
      ------------------statement
      ------------------statement
      ------------------if test-for-its-time-to-get-out-of-this-loop:
      ------------------break
      ------------------statement
      ------------------statement
19. What value must a variable used as a counter be initialized to?
   \[ \text{counterVar} = 0 \]
   When? before the loop
What is the update step for a counter variable?
   \[ \text{counterVar} += 1 \]
   When? inside the loop each time a new value is dealt with

20. Same questions for an accumulator
Initialization value \( \text{accumulatorVariable} = 0 \)
   When? before the loop
What is the update step for a counter variable?
   \[ \text{accumulatorVariable} += \text{currentValue} \]
   When? inside the loop each time a new value is gotten

21. A module must contain a main def.
   No module should have a main ever.
   Modules are like tool boxes, they do not solve any whole problem.
   main is where a whole problem is solved by calling other functions

22. A module must not do any imports
   Wrong – a module must do exactly as many imports as it needs to do it's jobs.
   It is NOT the responsibility of any other programmer to do the import for the module.