1. Consider the modular structure of a program to compose music for a band consisting of guitar, drum, and piano. Each instrument comes with \texttt{in} and \texttt{out} functions. The \texttt{in} function takes instructions and simulates the sound of the instrument. The \texttt{out} function prints instructions for the musician to play the composition.

We can design the program in two ways:

(a) There are two modules: \texttt{input} and \texttt{output}. The \texttt{input} module collects all \texttt{in} functions of the instrument. All \texttt{out} functions are contained in the module \texttt{output}.

(b) There is a module for each instrument. Each module contains the specific \texttt{in} and \texttt{out} functions for the instrument.

Which design would be best? Justify using the principles of good modular design.

Answer:

The second design works best.

Justification along the three principles of good modular design:

(a) information hiding: The programmer of a module in the first design needs to know the ins and outs of each instrument, while in the second design, a programmer of a module can focus on one instrument. In the second design, each module hides the details about its instrument to the other modules.

(b) high cohesion and low coupling: There is high cohesion in the second design because all functionality about one instrument is concentrated in one module, while the functionality of one instrument is spread out over several modules in the first design. In the first design there is high coupling because the instructions for the input to simulate the sounds will be similar to what will be given to the musicians. There is low coupling in the second design, because the programmer can share conventions in input and output routines, proper for each instrument.

(c) design for change: If another instrument is added to the band, in the second design we have only to add another module, the existing modules remain the same. In the first design, we need to change all existing modules.
2. Several procedures in the computer algebra software system Maple are visible to the user, but the license explicitly forbids to modify or distribute the code. Does this license conform to GNU GPL? Justify your answer.

**Answer:**

While some part of the code is open source, the license does not conform to GNU GPL, because GPL allows the modification and redistribution of source code.

3. An algorithm to process \( n \) Gigabytes of data has cost \( O(n^2) \). Suppose processing one Gigabyte takes 15 minutes. What is the largest number of Gigabytes that the algorithm can process within an hour? Justify your answer.

**Answer:** 2 Gigabytes

**Justification:** \( O(n^2) \) means that as we double the input size \( n \), the cost is multiplied by 4, and \( 4 \times 15 \text{ minutes} = 60 \text{ minutes} \).

4. Give Python code to define the following function:

```python
def GetValue():
    """
    Asks the user for an integer number. As long as the user input cannot
    be converted into an int, an error message is printed and the user is
    invited to try again. The function returns the int given by the user.
    """

    while True:
        a = raw_input(‘give an integer number : ’)

        try:
            x = int(a)

            return x

        except:
            print ’wrong input, please try again’
```

**Answer:**

```python
def GetValue():
    """
    Asks the user for an integer number. As long as the user input cannot
    be converted into an int, an error message is printed and the user is
    invited to try again. The function returns the int given by the user.
    """

    while True:
        a = raw_input(‘give an integer number : ’)

        try:
            x = int(a)

            return x

        except:
            print ’wrong input, please try again’
```
5. Write a program that prompts the user for the name of a Python script.
   The program counts all assignments and appends the lefthand side of each assignment to a list. After reading the entire script, the program writes the number of assignments and the list of all variables that occurred at the left of an assignment.
   The length of the list must equal the number of assignments, mind assignments to tuples, e.g.: \((a, b) = (0, 1)\). You may assume the script has at most one statement per line, but do not make any assumptions about spaces or tabs.

Answer:

```python
name = raw_input('give the name of the script : ')

file = open(name,'r')

P = []
cnt = 0

while True:
    s = file.readline()
    if s == '': break

    L = s.split(' ')

    if '=' in L:
        cnt = cnt + 1
        i = L.index('=')
        a = '' .join(L[0:i])
        P.append(a)

print '#assignments : ', cnt, 'arguments : ', P

file.close()
```
6. Draw the class diagram to represent a bank account. The name of the account is public, whereas the balance is protected by a password. Do not give Python code but explain how in Python certain object data attributes can be hidden.

Answer:

<table>
<thead>
<tr>
<th>Account</th>
</tr>
</thead>
<tbody>
<tr>
<td>name of account</td>
</tr>
<tr>
<td>_balance</td>
</tr>
<tr>
<td>_password</td>
</tr>
<tr>
<td>create/delete</td>
</tr>
<tr>
<td>show balance</td>
</tr>
<tr>
<td>modify balance</td>
</tr>
</tbody>
</table>

In Python, an object data attribute is made private by starting its name with an underscore. To hide the balance and password we use the names `_balance` and `_password`.

7. The RGB code of a color is a triplet of numbers between zero and one. Red is encoded as (1.0, 0.0, 0.0), Green as (0.0, 1.0, 0.0), and Blue as (0.0, 0.0, 1.0). Draw the design of a GUI to display colors corresponding to any RGB code \((r, g, b)\). Label each component of the GUI with the proper name of the widget. Do not give any Python code.

Answer:

We have used four labels, one to indicate the working of the GUI and three labels for the three scales. The three scales are used to enter the red, green, and blue intensities. Finally, the canvas displays the selected color.