GX1482

OLLSCOIL NA hÉIREANN GAILLIMH

NATIONAL UNIVERSITY OF IRELAND GALWAY

SEMESTER 1 EXAMINATIONS 2003

Third Arts University Examination in Information Technology
Fourth Arts University Examination in Information Technology

Object Oriented Programming (CT335)

Professor D. Bell
Professor G. Lyons
Dr. A. Golden

Time allowed: **Two hours**
Answer any three questions.

1. (a) Compare and contrast the object types lists, dictionaries and tuples in Python, giving examples of their notation, and when such object types might be appropriate.

   (b) The following is the result of an interactive session with the Python interpreter:

```
>>> class Message:
...     def __init__(self, string):
...         self.text = string
...     def print_out(self):
...         print self.text
...
>>> class Add_output(Message):
...     def print_out(self):
...         print 'OUTPUT = "%s"' % self.text
...
>>>
```

   - What is the relationship between `class Message` and `class Add_output`?
   - Explain the role of `self` in both class definitions.
   - How would you invoke an instance of the `class Message` from the command line? Calling the function `print_out()`: with that instance, what would the output be?
   - Similarly, how would you do this for `class Add_output`? What in effect would be happening in this case?

2. (a) What is meant by 'slicing' and 'indexing' for the object types strings, lists and tuples? Illustrate your answer with some examples for each case. Why are these operations inappropriate for dictionary object types?

   (b) In what way does object referencing in Python facilitate the use of functions? Is there a way to explicity communicate with a defined function?

3. (a) What is meant by a module file, why are they useful object types, and what is the relationship between any set of module files and the 'main' module file - what is the relationship between the names __main__ and __name__ in this regard? How are module files 'brought in' to running Python code?

   (b) Write brief notes on the `apply`, `map` and `lambda` built-in functions, illustrating your answer with suitable examples.

4. (a) What is the object oriented relationship between Classes, Instances, Subclasses and Superclasses? In what way is the term 'encapsulation' relevant to the relationship you've just defined?

   (b) What is meant by the term 'mutability' in the Python programming environment? How is this related to the idea of Python making references to object types with given names?

5. (a) How are methods and other class attributes for given instances of a class inherited? What is the syntax used to request use of a given inherited method - illustrate your answer with a suitable example for such a 'request'.

   (b) In Python, unlike other programming languages, there is no need to explicitly declare the object type for a nominated variable in advance of the main body of the code. How does Python deal with this?