Csci 1523 **Student (Print):** _____

1523 Study Guide Module 07 - Program Structure

This study guide contains 10 pages (including this cover page) and 40 problems. Check to see if any pages are missing. Enter all requested information on the top of this page, and put your initials on the top of every page, in case the pages become separated.

You may use your books, notes, calculator or internet sources while completing this study guide.

Please try to answer the sections clearly and PRINT your answers legibly.

Special Note: The completion of study guides in this course is to supplement your learning of the materials they are not required as a part of normal grading. However they may enhance the extra credit portion of the course if attendance has been regular and laboratories completed satisfactorily.

Chapter 7 Dierbach Study Guide
Creating large programs often involves the systematic decomposition of a problem into manageable units around which specific codes can be built. Once these units are defined and coded they can be systematically integrated into the main or driving program which solves the problem.

We observed this type of decomposition when in Module 05 in which we looked at functions at the lowest level of decomposition and the use of functions within a program to complete its development.

In this module we will extend the idea of functions to *modules* which are collections of related Python codes packaged into a separate file. These files can then be imported into the main program as needed and have the advantage of being able to be including in other programs as well.

Below you will find a series of questions concerning modules, functions, namespaces, top-down design and testing. The materials required to answer them are Chapter 7 of Dierbach, course notes and movies.

1. Modules generally consist of a collection of related _____ in Python.
2. Modules are in _____ files which can be imported by separate Python programs.
3. Python modules have a number of advantages in software development. Below are functional areas of development. In the space provided below each list the advantages modules bring to the software development process:
    1. **SOFTWARE DESIGN**

        .....
        .....
        .....
        .....
        .....
        .....
        .....
        .....
    2. **SOFTWARE DEVELOPMENT**

3. **SOFTWARE TESTING**

4. **SOFTWARE MODIFICATION AND MAINTENANCE**

4. The term _____ refers to the design and/or implementation of specific functionality to be incorporated into a program.

5. The term _____ refers to the specification provided for use of a given module. In Python, a _____ is used to provided such specification.

6. Any program making use of a particular module is refered to as a _____ of the module.

7. A _____ is a string literal denoted by triple qoutes given as the first line of certain program elements.

8. Suppose you wish to view a function's specification, assuming it is provided for a function, *factorial* in the space provided below write a single line of Python code which will display it to the terminal screen:

   . . . . . .
   . . . . . .
   . . . . . .
   . . . . . .
   . . . . . .
   . . . . . .
   . . . . . .
   . . . . . .

   Editorial note: The contents of doc strings which document how a function works can vary between development shops. It is usually good practice to include the following in doc strings:

   1. the function name
   2. the formal parameters along with their expected value ranges
   3. a description of the function's operations
   4. sample data and hand worked expected results, (for testing)
   5. return values and types
   6. description of any possible exceptions

   A properly prepared doc string enables a concept of programming to contract. In this case the doc string spells out the terms of such contracts. A properly prepared doc string enables a concept of programming to contract. In this case the doc string spells out the terms of such contracts.

9. A doc string can be thought of as the English language statement of the specification of the _____ to the function.

10. Select from the true statements from the list below concerning doc strings. A doc string is:

    A. A string literal denoted by triple of double qoutes.

    B. A means of providing specification for certain program elements in Python.

    C. A string literal that may span more than one line.

11. For the following function,

```
def hoursofdaylight(month, year)
```

    (a) In the space provided below give an appropriate docstring specification where `hoursofdaylight`, returns the total number of hours of daylight for the month

and year given (each passed an integer value) designed so that the function does not check for invalid parameter values.

(b) In the space provided below give a print statement that displays the docstring for this function.

12. _____ is a method for breaking down the overall design and then developing more detailed design aspects later.

13. Which of the following is characteristic of the top-down design methodology, choose all that apply:

    A. Provides a means for the development of well-designed programs.

    B. Provides a natural means of dividing up programming tasks.

    C. Provides a means of separately testing individual parts of a program.

14. A Python module is a **file** containing Python _____ and _____

15. The _____ is the module which is loaded and directly executed.

16. Main modules are given a special name in Python, which is _____.

17. ___ Main modules may import other modules and these may import other modules but a main module may not be imported.

18. The statements of imported modules are executed only once, when the module is _____

19. The Python Standard Library contains a set of predefined _____. We have used two of these the _____ and _____ modules.

20. The idea of modules is tightly related to the notion of a named context for a set of variables. We call this named context a _____.

21. In the space below describe what is meant by a *name clash*:

. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .

22. How do *namespaces* help prevent *name clashes*.

. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .

23. When working in the Python shell the shell serves as the main module. It contains the _____ namespace.

24. ___ The global namespace is reset every time the interpreter is started.

25. The module _____ is automatically imported into Python programs. It provides the built-in constants, functions and classes.

26. When using the: import `modulename` form of import the identifiers within the imported module must be _____ in order to be used within the client code.

27. When using the: import `modulename` form of import, the namespace becomes _____ but does not become _____ the namespace of the importing module.

28. The **from *module* import *something*** format for importing a module allows the namespace of the client to be combined with that of the imported module thereby eliminating the need to use fully qualified, *(modulename.identifier)*, identifiers to access members of the imported module. There are variants to this syntax. Below are three examples of this. Explain in detail the effect of each such variant.

1. **from *modulename* import *func1, func2***

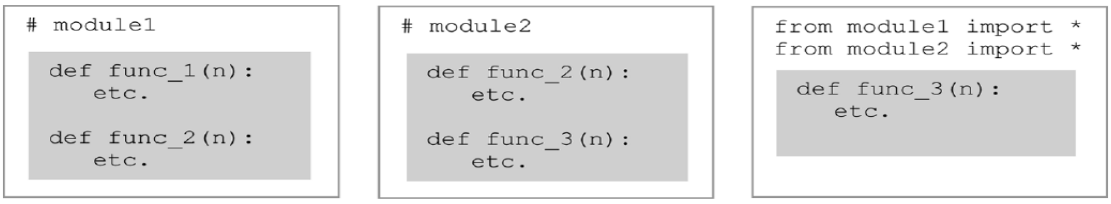
2. **from *modulename* import *func1 as localfunc1***

3. **from *modulename* import ***

29. Use of the **from *modulename* import *something*** form of import increases the liklehood of errors due to _____ between identifiers.

30. ___ It is a Python syntax error to import a builtin module prior to importing the user defined ones.

31. For module1, module2, and the client module shown below, indicate which of the imported identifiers would result in a name clash if the imported identifiers were not fully qualified. Identifier causing name clash: _____



32. Modules sometimes contain variables with no meaning outside the context of the module or are variables we don't wish to be changed outside of the module context. These variables are referred to as _____ variables.

33. ___ Python does not provide any mechanism to keep variables not meant to be accessed private.

34. ___ Python uses coding convention as a means of preventing variables which are meant to be accessed only within a module to not be accessed outside of it.

35. Below find a set of locations Python will search to include an imported module. Using numbers 1..4 place the order number adjacent to the location.

    1. _____ Python's installation specific path.
    2. _____ The client's local directory.
    3. _____ Directories found on the PYTHONPATH environment variable.

36. If an imported module is not found Python will cause an _____ exception to be thrown.

37. All Python applications have namespaces associated with them. Below find the built-in, global and local namespace listed. Describe their contents in the space provided:

    (a) **built-in**

    .....
    .....
    .....
    .....
    .....
    .....
    .....
    .....

    (b) **global**

(c) **local**

38. When Python references an identifier it searches the namespaces in a specified order. Using digits 1..3 specify the order in the space provided next to the namespaces listed below:
    (a) _____ built-in
    (b) _____ global
    (c) _____ local

39. For the following program and the imported modules, describe in the space provided below any name clashes that would occur for both program version1 and version 2.

```
# module m1

def total(items):

def convert(items):

def show(items)
```

```
# module m2

def totalSum(items):

def convert(items):

def display(items)
```

```
from m1 import *
from m2 import *

def display()

def calc()

def getItems()

# ---- main

items = getItems()
items = convert(items)
show(items)
```

Version 1

```
import m1
import m2

def display()

def calc()

def getItems()

# ---- main

items = getItems()
items = m2.convert(items)
display(items)
```

Version 2

40. ___ Errors resulting in name clashes are only found at runtime.