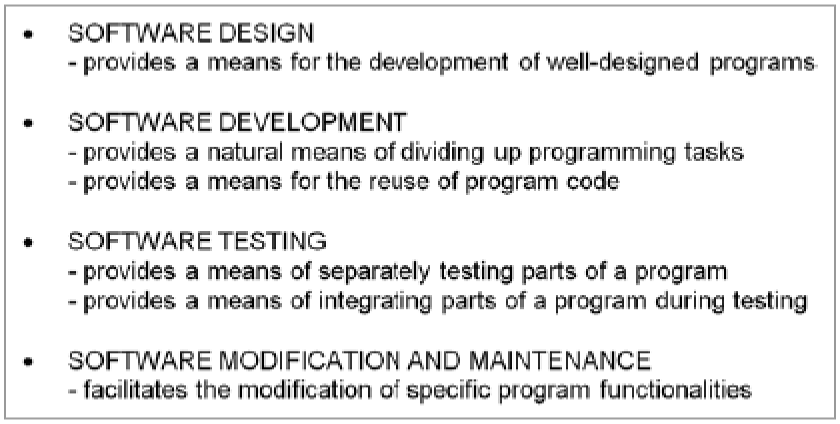
Modules

An important aspect of well-designed software is that programs are designed as a collection of modules. The term “module,” broadly speaking, refers to the design and/or implementation of specific functionality to be incorporated into a program. While an individual function may be considered a module, modules generally consists of a collection of functions (or other entities). The Python turtle module is an example of a software module. The use of modules has a number of advantages as shown below:

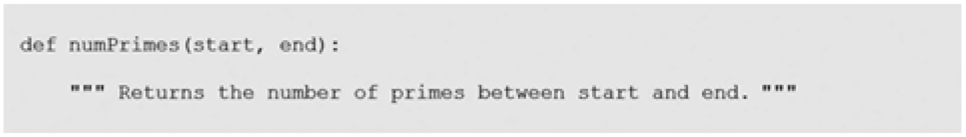


- SOFTWARE DESIGN
  - provides a means for the development of well-designed programs
- SOFTWARE DEVELOPMENT
  - provides a natural means of dividing up programming tasks
  - provides a means for the reuse of program code
- SOFTWARE TESTING
  - provides a means of separately testing parts of a program
  - provides a means of integrating parts of a program during testing
- SOFTWARE MODIFICATION AND MAINTENANCE
  - facilitates the modification of specific program functionalities

Modular design allows large programs to be broken down into manageable size parts, in which each part (module) provides a clearly specified capability. It aids the software development process by providing an effective way of separating programming tasks among various individuals or teams. It allows modules to be individually developed and tested, and eventually integrated as a part of a complete system. Finally, modular design facilitates program modification since the code responsible for a given aspect of the software is contained within specific modules, and not distributed throughout the program.

Module Specification

Every module needs to provide a *specification* of how it is to be used. This is referred to as the module’s **interface**. Any program code making use of a particular module is referred to as a **client** of the module. A module’s specification should be sufficiently *clear* and *complete* so that its clients can effectively utilize it. For example, `numPrimes` is a function that returns the number of primes in a given integer range, as shown below:



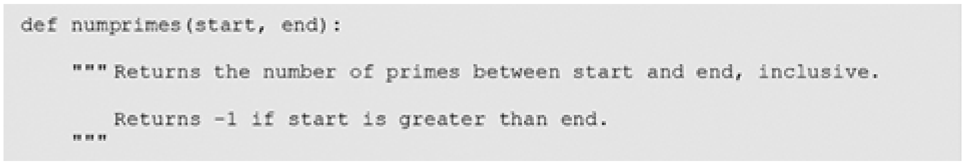
```
def numPrimes(start, end):

    """ Returns the number of primes between start and end. """
```

The function’s specification is provided by the line immediately following the function header, called a *docstring* in Python. A **docstring** is a string literal denoted by triple quotes given as the first line of certain program elements. The docstring of a particular program element can be displayed by use of the __doc__ extension,

```
>>> print(numPrimes. doc )
Returns the number of primes between start and end.
```

This provides a convenient way for discovering how to use a particular function without having to look at the function definition itself. Some software development tools also make use of docstrings. Let’s consider how complete this specification is for this function. At first look it may seem suffi- cient. However, it does not answer whether the number of primes returned includes the endpoints of the range or not. Also, it is not clear what will happen if the function is called with a first argument (`start`) greater than the second (`end`). Thus, a more complete specification is needed.



```
def numprimes(start, end):

    """ Returns the number of primes between start and end, inclusive.

        Returns -1 if start is greater than end.
    """
```

This is now a reasonable specification of the function. This docstring follows the Python convention of putting a blank line after the first line of the docstring, which should be an overall description of what the function does, followed by an arbitrary number of lines providing additional details. These additional lines must be indented at the same level, as shown in the figure. Appropriate use of this function by a client is given below.

```
.
.
first_num = int(input('Enter the start of the range: '))
second_num = int(input('Enter the end of the range: '))

    result = numprimes(first_num, second_num)
    if result == -1:
        print('* Invalid range entered *')
    else:

        print('The number of primes between', first_num, 'and',
           second_num, 'is', result)
```

In this example, the user inputs a start and end value for the range of integers to check. Since the user may inappropriately enter a start value greater than the value of the end value, a check is made for a returned value of `-1` after the call to `numprimes`. If `-1` is found, then an error message is output; otherwise, the result is displayed.

There are potential problems when returning both a computed result and an error result as a function’s return value. First, there is no guarantee that the client will perform the necessary check for the special error value. Thus, an incorrect result may be displayed to the user,

```
The number of primes between 100 and 1 is -1
```

Second, there may not be a special value that *can* be returned for error reporting. For example, if there is a function meant to return any integer value, including negative numbers, there is no special integer value that can be returned.

Concepts and Procedures

1. Which of the following is not an advantage in the use of modules in software development?

a) Provides a natural means of dividing up programming tasks.
b) Provides a means of reducing the size of a program.
c) Provides a means for the reuse of program code.
d) Provides a means of separately testing individual parts of a program.
e) Provides a means of integrating parts of a program during testing.
f) Facilitates the modification of specific program functionalities.

2. A specification of how a particular module is used is called the module's
.

3. Program code that makes use of a given module is called a______________ of the module.

4. Indicate which of the following are true. A `docstring` in Python is

a) A string literal denoted by triple or double quotes.
b) A means of providing specification for certain program elements in Python
c) A string literal that may span more than one line.

5. For the following function,

```
def hours_of_daylight(month, year)
```

a) Give an appropriate docstring specification where `hours_of_daylight` 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) Give a print statement that displays the docstring for this function.

Programming Problems

1. Write a function called `convertStatus` that is passed status code `'f'`, `'s'`, `'j'`, or `'r'` and returns the string `'freshman'`, `'sophomore'`, `'junior'`, or `'senior'`, respectively. Design your function so that if an inappropriate letter is passed, an error value is returned. Make sure to include an appropriate docstring with your function.

2. Write a function called `palindromeChecker` using iteration to return `True` if a provided string is a palindrome, and `False` otherwise. Make sure to include docstring specification for the function.