Chapter 16 Answers
In-Chapter Exercises

16.1
1.Integer Division by modulo or zero
2.6/18=0! Change the 6 to a 6.0.
3.7
4.Remainder or modulo

16.2
True, all relations are correct
False, 6 is not greater than 71
0, What’s in ( ) is evaluated first, thus zero(False) divided by anything equal 0
0, What’s in ( ) is evaluated first, thus zero(False) divided by anything equal 0

16.3
1. x is something else
2. Nothing prints out
3. x is something else

16.4
1. The error reads- RuntimeError: maximum recursion depth exceeded in cmp. cmp is a python function that discriminates to only allow integers forward in the calculations. You can’t use a Real number to find its factorial!
2. Use the gamma function on the Real number + 1, in our case10.76. Such as-

```
>>>import math
>>>math.gamma(10.76)
2069368.0609079208
```

16.5
1. >>>print int(x1[0]) + int(x1[1]) + int(x1[2])
2. newline
3. You can open the file, read from it, and print x1, but x1 will have a single object, 23 33 43 54\n in it, basically a character string of 4 numbers and a newline, and none of the rest of the commnds in the Example will work on that object!
4. Yes, by specifying the objects as int(x1[2]) and int(x1[3]).
5. x1[2]

16.6
1. programmingprogrammer
programs
2. newline is a character
3. 0
4. Empty set
5. Extracts the whole string.
6. ‘ro’ because 1:3 extracts from the second character to the fourth character, <u>not including the fourth.</u>

16.7
1. From the assignments and operations above that line, there are no elements common to r and a.

2. x.add takes only one argument. x.update will only add one 5.0.
3. No, it is an immutable object. For example-

```
>>> s = (34,35,36,37)
>>> s[1:2] = [-1,-2]
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    s[1:2] = [-1,-2]
TypeError: 'tuple' object does not support item assignment
>>>
```

16.8

```
1. >>>import regen
2. >>>x = regen.abc()
3. >>>x.next()
1
>>>x.next()
2
>>>x.next()
3
>>>x.next()
‘ijk’
>>>x.next()
‘deff’
>>>x.next()
‘abc’
>>>x.next()
Traceback(most recent call last)
File “<stdin>”, line 1, in <module>
StopIteration
>>>
```

16.9

1. a) in a file named grep.py-

```
def grepper(template):
    print(‘Searching for ‘ + template)
    try:
        while True:
            x = (yield)
            id template in x:
                print x
    except GeneratorExit:
        print “Done”
```

b)

```
>>>import grep
>>>s = grep.grepper(‘1201’)
```

c)

```
>>>s.next()
Searching for 1201
>>>s.send(‘0120 1201 1020’)
```

```
0120 1201 1020
>>>s.send('3012 3013 3212')
>>>s.send('12010203101213012')
12010203101213012
>>>
```
2. a)
```
def read(text,next_coroutine):
     for line in text.split():
          next_coroutine.send(line)
    next_coroutine.close()
```
b) assumes grepper module or function is still loaded!
```
>>>import reader
>>>text2 = " Python is the most Pythonic enterprise a Pythonista can practice"
>>>found2 = grep.grepper("Python")
```
c)
```
>>>found2.next()
Searching for Python
>>>reader.read(text2,found2)
Python
Pythonic
Pythonista
Done
>>>
```

16.10
1. SystemExit – sys.exit() function
ArithmeticError – numeric calculations error
EOFError – end of file reached
ImportError – import statement error
KeyError – dictionary key not found
2. Add a new line 3, L = [1,2,3,4,5,6,7,8,9,10]
On new line 4, handler.write(str(L))

16.11
1. Traceback NameError name 'b' is not defined, b is a local variable in the module arith1.
2. Traceback NameError name 'arith1' is not defined, del has erased the module from Python memory.
3. You get no values, because the functions are not bringing to or associating values with the named objects at the Python comman line level(which can be called the top level, or the highest module).

16.12
1. /bin/sh name , where name is the file that contains the Bourne code.
2. os.path.exists(path)
3. Assuming no one has a /usr/bin/yyy directory, the else statement is executed.

16.13
1. words[0], words[1], words[2:]
2. r.split(' ') , ' '.join(words

16.14
1. os.popen('umount -f /dev/da1s1')
2. ufs is a common one, substitute ufs for msdosfs in steps 7. and 10.
3. Nothing, if you followed the 11 steps in Example 16.26. No.
4. No answer required.

16.15 No answer required.

16.16
1. Methods are basically an OOP operation on objects, classes, and instances, and functions are modularized, callable-code that declare a procedure to perform in a non-OOP manner, such as in a procedural/functional programming language like C.
2. Numbering starts in the upper left hand corner at 0,0, and proceeds down rows and to the left in columns. So the grid position 0,2 would translate to the zeroth(or top-most) row, the third column to the right.
3. Similar to the quit function shown in Example 16.34, you can import sys, and place a Button in grid cell 0,2 that calls sys.exit().

16.17. In the function child, after the print statement.

16.18. This illustrates the most important aspect of running threads concurrently, whether in Python or at the system programming level as shown in Chapters 19. There is no guarantee that, because of the scheduling of threads in the parent process, that the order of the **myId** variable will be printed exactly the same on each loop. That is the reason you run them concurrently.

16.19. Not on our PC-BSD system. Same reason as the answer to 16.18.

16.20 In Block 3., in the for loop before the **time.sleep(GoToSleep)** line.

16.21 MAX_NUM is the limit to the size of buffer, there to ensure no buffer overrun.

16.22 Because the scheduling of the output to stdout from the 2 threads is not entirely determinate, sometimes Producer runs for awhile, and then Consumer runs, etc.. This is an important characteristic of multi-threaded programs.