ANSWER KEY

1. (3 pts) The list type is an iterable object.

(circle your answer) TRUE FALSE

WHAT IS THE OUTPUT questions – write your answer to the right of the question

If the code won't compile, write "won't compile"
If it will raise a runtime exception, state the exception.
if you cannot remember the name of the exception, just write 'exception'

2. (3 pts)
```
one = "two"
two = "one"
two += one + "two"
print one, two
```
two onetwotwo

3. (3 pts)
```
lissy = { "testing" : "lissy" }
print( lissy [ "testing" ] )
```
lissy

4. (3 pts)
```
s = [ ['a'], ['b'], 'c' ]
t = s.pop( s.remove( 'a' ) )
print( t )
```
ValueError

5. (3 pts)
```
one = [ 2, 3, 4, 5, 6, 7 ]
one[ 2:4 ] = [ 55, 11 ]
print( one )
```
[2, 3, 55, 11, 6, 7]

WHAT IS THE OUTPUT questions – write your answer to the right of the question

If the code won't compile, write "won't compile"
If it will raise a runtime exception, state the exception.
if you cannot remember the name of the exception, just write 'exception'

6. (3 pts)

```
one = ( 2, 3, 4 )
print( one[ 2: ] )
```

(3, 4)

7. (3 pts)

```
one = [ 'snake', 'piggy', 'tiger'  ]
for two in one:
    print( two[ 2 ] )
```

a
g
g

8. (3 pts)

```
alpha = { 1:"joe", 3:"sue", "joe":4 }
beta  = { alpha[ "joe" ] : 1 }
print( beta )
```

{ 4 : 1 }

9. (**9 pts**) Given:

```
def alpha( one ):
    ''' ??? '''
    one.append( 3 )
    one = 3

def main():
    one = [ 4, 3, 2 ]
    print( "before" )
    print( one )
    alpha( one )
    print( "after" )
    print( one )
```

before
[4, 3, 2]
after
[4, 3, 2, 3]

When this program is run, what is the output?

WHAT IS THE OUTPUT questions – write your answer to the right of the question

If the code won't compile, write "won't compile"
If it will raise a runtime exception, state the exception.
if you cannot remember the name of the exception, just write 'exception'

10. (3 pts)

```
alpha = [ 4, '44', '4', [ '4' ], 'four' ]
alpha.remove( '4' )
print( alpha
```

[4, '44', ['4'], 'four'])

11. (3 pts)

```
alpha = { "S" : "snakey", "T" : "turkey", "D" : "doggie" }
print( alpha[ '''S''' ] )
```

snakey

12. (3 pts)

```
alpha = [ 5, 4, 3, 2, 1 ]
print( alpha.index(5)
```

0)

13. (3 pts) The dictionary type has the `.count()` method? (circle your one answer)

a) yes b) no

14. (15 pts) Complete the definition of this function so that it does what the docstring states.

```
def avgDepth( dictOfLakeToDepth, listOfLakeNames ):
    ''' Takes a dictionary of lake names that map to the lake's depth.
        The depth is of float type.
        Also takes a list of lake names to process.
        Returns the average depth of those lakes in the list of names.
        You are guaranteed that there is at least one of the lake
        names in both the dictionary and the list thus preventing
        division by zero from happening.
    '''
```

```
    sum = 0.0
    count = 0
    for name in listOfLakeNames:
        if name in dictOfLakeToDepth:
            sum += dictOfLakeToDepth[ name ]
            count += 1
    return sum / count
```

could write:

```
    for ndx in range( len( listOfLakeNames ) ):
        if listOfLakeNames[ ndx ] in dictOfLakeToDepth:
            sum += dictOfLakeToDepth[ listOfLakeNames[ ndx ] ]
```

but write: **"why use indexing?"**

Could also write:

```
    for key in dictOfLakeToDepth:
        if key in listOfLakeNames:
            sum += dictOfLakeToDepth[ key ]
```

There could be other acceptable ways.
Read carefully and let the other graders know about other ways and all agree that it is acceptable.

If they write code to check for exceptions and such, write "Why – will be one name in both".

15. (20 pts) Complete the definition of this function so that it does what the docstring states.

```
def removeAllButLast( listy, valueToRemove ):
    ''' Takes a list and a value to remove from the list.
        Returns a list with all but the last (rightmost) instances of
        value removed.
        Does not change the original that was passed in.
        For example if the passed in list is: [ 2, 4, 2, 6, 7, 2, 8 ]
        and 2 is to be removed,
                  the returned list will be: [ 4, 6, 7, 2, 8 ]
 '''

      import copy
      lst = copy.deepcopy( listy )

      qnt = listy.count( valueToRemove )


      for i in range( qnt – 1 ):
           lst.remove( valueToRemove )
---------OR ---------------
      for item in lst:
           if item == valueToRemove and qnt > 1:
                lst.remove( valueToRemove )
                qnt -= 1
---------OR other ways

      return lst
```