Introduction to GIS Programming and Fundamentals with Python and ArcGIS 1st edition Yang Solutions Manual

SKU: 9781466510081-SOLUTIONS

```
>>> # Fibonacci series:
>>>
>>>
>>>
>>> a, b = 0, 1
>>> while b < 10:
        print b
        a, b = b, a+b


1
1
2
3
5
8
>>> i = 256*256
>>> print 'the value of i is', i
the value of i is 65536
>>> a, b = 0, 1
>>> while b < 1000:
        print b
        a, b = b, a+b


1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
```

Chapter 2

1. Pick 3 points, e.g., (1,100), (25,60) and (1, 1). Could you form a polyline or polygon using these three points?

Three points compose a polyline, and four points compose a polygon with the first and last the same.

2. Create an algorithm to calculate the distance between two points, e.g., (x1, y1), (x2, y2).

$$distance = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$

3. Read Python Tutorial 6.2 and 6.3. (Use Python command line window for 6.2).

For 6.2, command line window is needed. For 6.3, we need to read and try from beginning of Tutorial 6.

```
>>> import sys
>>> sys.ps1
'>>> '
>>> sys.ps2
'... '
>>> sys.ps1 = 'C> ';
  File "<stdin>", line 1
    sys.ps1 = 'C> ';
                  ^
SyntaxError: EOL while scanning string literal
>>> sys.ps1 = 'C> '
C> print 'Yuck!'
Yuck!
C> import sys
C> sys.path.append('/ufs/guido/lib/python')
C> import fibo, sys
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: No module named fibo
C> import fibo, sys
C> dir(fibo)
['__builtins__', '__doc__', '__file__', '__name__', '__package__', 'fib', 'fib2']
C> dir(sys)
['__displayhook__', '__doc__', '__excepthook__', '__name__', '__package__', '__stderr__', '__stdin__', '__stdout__', '_clear_type
_cache', '_current_frames', '_getframe', '_mercurial', 'api_version', 'argv', 'builtin_module_names', 'byteorder', 'call_tracing'
, 'callstats', 'copyright', 'displayhook', 'dllhandle', 'dont_write_bytecode', 'exc_clear', 'exc_info', 'exc_type', 'excepthook',
 'exec_prefix', 'executable', 'exit', 'flags', 'float_info', 'float_repr_style', 'getcheckinterval', 'getdefaultencoding', 'getfi
lesystemencoding', 'getprofile', 'getrecursionlimit', 'getrefcount', 'getsizeof', 'gettrace', 'getwindowsversion', 'hexversion',
'last_traceback', 'last_type', 'last_value', 'long_info', 'maxint', 'maxsize', 'maxunicode', 'meta_path', 'modules', 'path', 'pat
h_hooks', 'path_importer_cache', 'platform', 'prefix', 'ps1', 'ps2', 'py3kwarning', 'setcheckinterval', 'setprofile', 'setrecursi
onlimit', 'settrace', 'stderr', 'stdin', 'stdout', 'subversion', 'version', 'version_info', 'warnoptions', 'winver']
C> import __builtin__
C> dir(__builtin__)
['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException', 'BufferError', 'BytesWarning', 'DeprecationWarning', 'EO
FError', 'Ellipsis', 'EnvironmentError', 'Exception', 'False', 'FloatingPointError', 'FutureWarning', 'GeneratorExit', 'IOError',
 'ImportError', 'ImportWarning', 'IndentationError', 'IndexError', 'KeyError', 'KeyboardInterrupt', 'LookupError', 'MemoryError',
 'NameError', 'None', 'NotImplemented', 'NotImplementedError', 'OSError', 'OverflowError', 'PendingDeprecationWarning', 'Referenc
eError', 'RuntimeError', 'RuntimeWarning', 'StandardError', 'StopIteration', 'SyntaxError', 'SyntaxWarning', 'SystemError', 'Syst
emExit', 'TabError', 'True', 'TypeError', 'UnboundLocalError', 'UnicodeDecodeError', 'UnicodeEncodeError', 'UnicodeError', 'Unico
deTranslateError', 'UnicodeWarning', 'UserWarning', 'ValueError', 'Warning', 'WindowsError', 'ZeroDivisionError', '_', '__debug__
', '__doc__', '__import__', '__name__', '__package__', 'abs', 'all', 'any', 'apply', 'basestring', 'bin', 'bool', 'buffer', 'byte
array', 'bytes', 'callable', 'chr', 'classmethod', 'cmp', 'coerce', 'compile', 'complex', 'copyright', 'credits', 'delattr', 'dic
t', 'dir', 'divmod', 'enumerate', 'eval', 'execfile', 'exit', 'file', 'filter', 'float', 'format', 'frozenset', 'getattr', 'globa
ls', 'hasattr', 'hash', 'help', 'hex', 'id', 'input', 'int', 'intern', 'isinstance', 'issubclass', 'iter', 'len', 'license', 'lis
t', 'locals', 'long', 'map', 'max', 'memoryview', 'min', 'next', 'object', 'oct', 'open', 'ord', 'pow', 'print', 'property', 'qui
t', 'range', 'raw_input', 'reduce', 'reload', 'repr', 'reversed', 'round', 'set', 'setattr', 'slice', 'sorted', 'staticmethod', '
str', 'sum', 'super', 'tuple', 'type', 'unichr', 'unicode', 'vars', 'xrange', 'zip']
C>
```

4. Define and program three classes for Point, Polyline, and Polygon.

```
import math
class Point:  ## define a point class
        def __init__(self, x=0.0, y=0.0):
                self.x = x
                self.y = y
        def calDis(self,p):
                return math.sqrt((p.x-self.x)**2+(p.y-self.y)**2)

class Polyline:  ## define a polyline class
        def __init__(self, points = []):
                self.points = points
        def getLength(self):
                length = 0.0
                for i in range(len(self.points)-1):
                        length += math.sqrt((self.points[i].x-self.points[i+1].x)**2+
                                            (self.points[i].y-self.points[i+1].y)**2)
                return length

class Polygon:  ## define a polygon class
        def __init__(self, points = []):
                self.points = points
        def getLength(self):
                length = 0.0
                for i in range(len(self.points)-1):
                        length += math.sqrt((self.points[i].x-self.points[i+1].x)**2+
                                            (self.points[i].y-self.points[i+1].y)**2)
                return length
```

5. Add distance calculation in-between every two points, and program to calculate the distance among the three points given.

```
>>> p1 = Point(1,100)
>>> p2 = Point(25,60)
>>> p3 = Point(1,1)
>>> p1.calDis(p2)
46.647615158762406
>>> p2.calDis(p3)
63.694583757176716
>>> p1.calDis(p3)
99.0
```

6. Add the getLength() method in Polyline and Polygon; create a polyline and polygon using the three points given; calculate the length of the polyline and perimeter of the polygon.

```
>>> pointList = [p1,p2,p3]
>>> polyline = Polyline(pointList)
>>> print polyline.getLength()
110.342198916
```

```
>>> pointList2 = [p1,p2,p3,p1]
>>> polygon = Polygon(pointList2)
>>> print polygon.getLength()
209.342198916
```

Chapter 3

1. Key words: Check the following key words (Table 1) and briefly explain them (concepts, when/how to use key words (use function help() to get help about each key word, e.g., help('if')), and design/program an example of how to use the keywords).

Example answers: