Claudiordgz

Solutions of Data Structures and Algorithms in Python

Book Authors: Goodrich, Tamassia and Goldwasser

Contents

Format

All exercises will be presented with their own Python Doctest documentation to allow testing. To run them in your own python package you can copy paste the text and add a main like the following:

Running Doctest

```
1 if __name__ == "__main__":
2     import doctest
3     doctest.testmod()
```

This is just to try to keep it as simple as possible while adding how to run the code in your own work environment.

Pro-Tip. JetBrains Pycharm is awesome, I really recommend it, plus they got a Community Edition if you are pennyless like me. The colors, the functionality it just rocks.

Pro-Tip. I like to use Anaconda for my Python distro, but the standalone Python 2.7 or >= 3 works too.

Python Primer

The first chapter in the book is all about learning to handle Python syntax. Subjects include objects, control flow, functions, I/O operations, exceptions, iterators and generators, namespaces, modules, and scope. There is nothing regarding python packaging to redistribute your own module, which is a subject of its own.

2.0.1 Exercises

The exercises in the first chapter are fun, no joke. I've seen what's coming in chapter 2 and those exercises look terrible because they are open ended questions, but they are also important concepts.

Go to Top

R-1.1

Write a short Python function, `is_multiple(n, m)`, that takes two integer values and returns `True` if n is a multiple of m, that is, $n = mi$ for some integer i, and `False` otherwise.

Exercise R-1.1

```
def is_multiple(n, m):
    """Return True if n is multiple of m
    such that n = mi else returns False

    >>> is_multiple(50,3)
    False
    >>> is_multiple(60,3)
    True
    >>> is_multiple(70,3)
    False
    >>> is_multiple(-50,2)
    True
    >>> is_multiple(-60,2)
    True
    >>> is_multiple("test",10)
    Numbers must be Integer values
    >>> is_multiple(-60,"test")
```

```
18      Numbers must be Integer values
19      """
20      try:
21          return True if (int(n) % int(m) == 0) else False
22      except ValueError:
23          print("Numbers must be Integer values")
```

Go to Top

R-1.2

Write a short Python function, `is_even(k)`, that takes an integer value and returns `True` if k is even, and `False` otherwise. However, your function cannot use the multiplication, modulo, or division operators.

Exercise R-1.2

```
1  def is_even(k):
2      """Return True if n is even
3      else returns False
4
5      >>> is_even(10)
6      True
7      >>> is_even(9)
8      False
9      >>> is_even(11)
10     False
11     >>> is_even(13)
12     False
13     >>> is_even(1025)
14     False
15     >>> is_even("test")
16     Number must be Integer values
17     """
18     try:
19         return int(k) & 1 == 0
20     except ValueError:
```

```
21            print("Numbers must be Integer values")
```

Go to Top

R-1.3

Write a short Python function, `minmax(data)`, that takes a sequence of one or more numbers, and returns the smallest and largest numbers, in the form of a tuple of length two. Do not use the built-in functions `min` or `max` in implementing your solution.

Exercise R-1.3

```
1  class MinMax():
2      def __init__(self, field1, field2):
3          self.min = field1
4          self.max = field2
5      def __str__(self):
6          return "Min {min} - " \
7                  "Max {max}".format(min=str(self.min),
8                                     max=str(self.max))
9
10 def minmax(data):
11     start = 0
12     mm = MinMax(data[start],data[start])
13     if len(data) & 1  == 1:
14         if data[start] < data[start+1]:
15             mm.max = data[start+1]
16             mm.min = data[start]
17             start += 2
18         else:
19             start += 1
20     for index in range(start, len(data[start:]), 2):
21         if data[index] < data[index+1] :
22             l_min = data[index]
23             l_max = data[index+1]
24         else:
25             l_min = data[index+1]
```

```
26              l_max = data[index]
27          if mm.min > l_min:
28              mm.min = l_min
29          if mm.max < l_max:
30              mm.max = l_max
31      return mm
```