Lab 10: Python

ESS 112 - Programming I

International Institute of Information Technology – Bangalore

Submission: LMS and Domjudge by 31 Jan 23:59:59

Submission Instructions: Submit your code in a separate file on LMS as well as Domjudge.

Note: You must implement all methods specified in the problem. You may additionally define additional methods if needed to implement the behaviour specified.

Exercises:

1. Implement a class to represent a complex number in python with the following methods:

```
def __init__(self, real, imag):
    """Initializes a complex number with real and imaginary parts"""

def __str__(self):
    """Returns a string representation of the complex number"""

def __add__(self, other):
    """Returns a complex number with addition of both complex number"""

def __sub__(self, other):
    """Returns a complex number with subtraction of both complex number"""

def __mul__(self, other):
    """Returns a complex number with multiplication of both complex number"""
```

Sample Input:

2 3
4 5

Sample Output:

2 + 3i
4 + 5i
6 + 8i
-2 - 2i
-7 + 22i

2. Implement a simple Bank Account class with deposit and withdrawal methods:

```
def __init__(self, balance=0):
    """Initializes a BankAccount object with a balance (default is 0)"""
```

```
def deposit(self, amount):
    """Deposits an amount into the bank account"""

def withdraw(self, amount):
    """Withdraws an amount from the bank account"""
```

Sample Input:
3
Deposit 50
Withdraw 35
Balance

Sample Output:
15

3. Implement a simple Rectangle class with width and height attributes. Create methods to calculate area and perimeter of rectangle.

```
def __init__(self, width, height):
    """Initializes a Rectangle object with a width and a height"""


def area(self):
    """Returns the area of the rectangle"""


def perimeter(self):
    """Returns the perimeter of the rectangle"""
```

Sample Input:
10 5

Sample Output:
50
30

4. Implement a simple stack class in Python with the following methods:

```
def __init__(self):
    """Initializes an empty stack"""

def push(self, item):
    """Inserts an item at the top of the stack"""
```

```
  def pop(self):
    """Removes and returns the item at the top of the stack"""

  def is_empty(self):
    """Returns True if the stack is empty, False otherwise"""

  def __str__(self):
    “””Returns a string representation of the stack”””
```

Sample Input:

5
Push 1
Push 2
Output
Pop
Output

Sample Output:

2 1
1

5. Implement a binary search tree class in Python with the following methods and classes: (Bonus question – optional)

   A tree is a data structure composed of nodes that has the following characteristics:

   a. Each tree has a root node at the top (also known as Parent Node) containing some value (can be any datatype).
   b. The root node has zero or more child nodes.
   c. Each child node has zero or more child nodes, and so on. This creates a subtree in the tree. Every node has its own subtree made up of its children and their children, etc. This means that every node on its own can be a tree.

   A binary search tree (BST) adds these two characteristics:

   d. Each node has a maximum of up to two children.
   e. For each node, the values of its left descendent nodes are less than that of the current node, which in turn is less than the right descendent nodes (if any).

```
class Node:
  def __init__(self, data):
    """Initializes a Node object with data and left and right children set to None"""

class BST:
  def __init__(self):
    """Initializes a BST object with the root set to None"""

  def insert(self, data):
    """Inserts a node with data into the BST"""
```

```
    def find(self, data):
        """Find a node with data into the BST"""
```

Sample Input:

```
5
Insert 1
Insert 2
Insert 3
Find 1
Find 4
```

Sample Output:

```
True
False
```