Abstract Data Type using Object-Oriented Programming in Python

Abstraction

Abstraction is one of the most powerful ideas in computer science. It separates the what from the how.

Abstraction provides modularity.

Classes are the Python representation for “Abstract Data Types,” (ADT) a very useful notion in any programming language.

An ADT involves both data and operations on that data.

Example: Stack

A stack is a LIFO (last in, first out) list with the following operations: Push, Pop, Create (Init), Top, IsEmpty, Size. (Such a list is called a Signature or the Interface to the ADT.)

Stack ADT

Stack() creates a new stack that is empty. It needs no parameters and returns an empty stack.
push(item) adds a new item to the top of the stack. It needs the item and returns nothing.
pop() removes the top item from the stack. It needs no parameters and returns the item. The stack is modified.
top() returns the top item from the stack but does not remove it. It needs no parameters. The stack is not modified.
isEmpty() tests to see whether the stack is empty. It needs no parameters and returns a boolean value.
size() returns the number of items on the stack. It needs no parameters and returns an integer.

Stack ADT as a Class

```
class Stack:
  def __init__(self):
      self.items = []
  def IsEmpty(self):
    return self.items == []
  def push(self,item):
    self.items.append(item)
  def pop(self):
    return self.items.pop()
  def top(self):
    return self.items[len(self.items)-1]
  def size(self):
    return len(self.items)
```

Example:Queue

A Queue is a FIFO (first in, first out) list with the following operations: Enqueue, Dequeue, Size, Font.

Queue ADT

Queue() creates a new queue that is empty. It needs no parameters and returns an empty queue.
Enqueue(item) adds a new item to the rear of the queue. It needs the item and returns nothing.
Dequeue() removes the item from the front of the queue. It needs no parameters and returns the item. The queue is modified.
Front() returns the front item from the queue but does not remove it. It needs no parameters. The queue is not modified.
isEmpty() tests to see whether the queue is empty. It needs no parameters and returns a boolean value.
size() returns the number of items on the queue. It needs no parameters and returns an integer.

Queue ADT as a Class

```
Class Queue:
  def __init__(self):
      self.items = []
  def IsEmpty(self):
    return self.items == []
  def enqueue(self,item):
    self.items.append(item)
  def dequeue(self):
    return self.items.pop(0)
  def front(self):
    return self.items[len(self.items)-1]
  def size(self):
    return len(self.items)
```

Rational ADT as a Class

```
class Rational:
   def __init__(self,n,d):
       if d==0:
         print 'The denominator is ZERO, can not proceed'
       else:
         g=gcd(n,d)
         self.numer=n/g
         self.denom=d/g
   def show(self):
      print self.numer,'/',self.denom
   def add(self,a):
      n=self.numer*a.denom + self.denom*a.numer
      d=self.denom*a.denom
      return Rational(n,d)
```

Rational ADT as a Class

```
def sub(self,a):
    n=self.numer*a.denom - self.denom*a.numer
    d=self.denom*a.denom
    return Rational(n,d)
def mult(self,a):
    n=self.numer*a.numer
    d=self.denom*a.denom
    return Rational(n,d)
def divide(self,a):
    n=self.numer*a.denom
    d=self.denom*a.numer
    return Rational(n,d)
def equal(self,a):
    a1=self.numer*a.denom
    a2=self.denom*a.numer
    return (a1==a2)
```