fraction.py (Page 1 of 2)

```
 1: #!/usr/bin/env python
 2:
 3: #-----------------------------------------------------------------------
 4: # fraction.py
 5: # Author: Bob Dondero
 6: #-----------------------------------------------------------------------
 7:
 8: import euclid
 9:
10: #-----------------------------------------------------------------------
11:
12: class Fraction:
13:
14:     def __init__(self, num=0, den=1):
15:         if den == 0:
16:             raise ZeroDivisionError('Denominator cannot be 0')
17:         self._num = num
18:         self._den = den
19:         self._normalize()
20:
21:     def _normalize(self):
22:         if self._den < 0:
23:             self._num *= -1
24:             self._den *= -1
25:         if self._num == 0:
26:             self._den = 1
27:         else:
28:             gcden = euclid.gcd(self._num, self._den)
29:             self._num //= gcden
30:             self._den //= gcden
31:
32:     def __str__(self):
33:         if self._den == 1:
34:             return str(self._num)
35:         return '%d/%d' % (self._num, self._den)
36:
37:     def __hash__(self):
38:         return hash((self._num, self._den))   # Use tuple's __hash__().
39:
40:     def __eq__(self, other):
41:         return (self._num == other._num) and (self._den == other._den)
42:
43:     def __ne__(self, other):
44:         return not self == other
45:
46:     def __lt__(self, other):
47:         return (self._num * other._den) < (other._num * self._den)
48:
49:     def __gt__(self, other):
50:         return (self._num * other._den) > (other._num * self._den)
51:
52:     def __le__(self, other):
53:         return not self > other
54:
55:     def __ge__(self, other):
56:         return not self < other
57:
58:     def __neg__(self):
59:         return Fraction(-self._num, self._den)
60:
61:     def __add__(self, other):
62:         new_num = (self._num * other._den) + (other._num * self._den)
63:         new_den = self._den * other._den
64:         return Fraction(new_num, new_den)
65:
```

fraction.py (Page 2 of 2)

```
66:     def __sub__(self, other):
67:         new_num = (self._num * other._den) - (other._num * self._den)
68:         new_den = self._den * other._den
69:         return Fraction(new_num, new_den)
70:
71:     def __mul__(self, other):
72:         new_num = self._num * other._num
73:         new_den = self._den * other._den
74:         return Fraction(new_num, new_den)
75:
76:     def __truediv__(self, other):
77:         new_num = self._num * other._den
78:         new_den = self._den * other._num
79:         return Fraction(new_num, new_den)
```

fractionclient.py (Page 1 of 2)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# fractionclient.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import sys
import fraction

def main():

    try:

        line = input('Numerator 1: ')
        num1 = int(line)
        line = input('Denominator 1: ')
        den1 = int(line)
        line = input('Numerator 2: ')
        num2 = int(line)
        line = input('Denominator 2: ')
        den2 = int(line)

        frac1 = fraction.Fraction(num1, den1)
        print('frac1:', str(frac1))  # Same as frac1.__str__()

        frac2 = fraction.Fraction(num2, den2)
        print('frac2:', frac2)  # print() calls str(frac2)
                                # Same as frac2.__str__()

        print('frac1 hashcode:', hash(frac1)) # Same as frac1.__hash__()

        if frac1 == frac2:  # Same as frac1.__eq__(frac2)
            print('frac1 equals frac2')
        if frac1 != frac2:  # Same as frac1.__ne__(frac2)
            print('frac1 does not equal frac2')
        if frac1 < frac2:  # Same as frac1.__lt__(frac2)
            print('frac1 is less than frac2')
        if frac1 > frac2:  # Same as frac1.__gt__(frac2)
            print('frac1 is greater than frac2')
        if frac1 <= frac2:  # Same as frac1.__le__(frac2)
            print('frac1 is less than or equal to frac2')
        if frac1 >= frac2:  # Same as frac1.__ge__(frac2)
            print('frac1 is greater than or equal to frac2')

        frac3 = -frac1  # Same as frac1.__neg__()
        print('-frac1:', frac3)

        frac3 = frac1 + frac2  # Same as frac1.__add__(frac2)
        print('frac1 + frac2:', frac3)

        frac3 = frac1 - frac2  # Same as frac1.__sub__(frac2)
        print('frac1 - frac2:', frac3)

        frac3 = frac1 * frac2  # Same as frac1.__mul__(frac2)
        print('frac1 * frac2:', frac3)

        frac3 = frac1 / frac2  # Same as frac1.__truediv__(frac2)
        print('frac1 / frac2:', frac3)

    except Exception as ex:
        print(str(ex), file=sys.stderr)

#-----------------------------------------------------------------------

```

fractionclient.py (Page 2 of 2)

```
if __name__ == '__main__':
    main()
```

EqualityStr.java (Page 1 of 1)

```
 1: import java.util.Scanner;
 2:
 3: public class EqualityStr
 4: {
 5:    public static void main(String[] args)
 6:    {
 7:       Scanner scanner = new Scanner(System.in);
 8:
 9:       System.out.println("Enter the first string:");
10:       String s1 = scanner.nextLine();
11:
12:       System.out.println("Enter the second string:");
13:       String s2 = scanner.nextLine();
14:
15:       System.out.println(s1 == s2);
16:       System.out.println(s1.equals(s2));
17:    }
18: }
```


equalitystr.py (Page 1 of 1)

```
 1: #!/usr/bin/env python
 2:
 3: #-----------------------------------------------------------------------
 4: # equalitystr.py
 5: # Author: Bob Dondero
 6: #-----------------------------------------------------------------------
 7:
 8: def main():
 9:
10:     s1 = input('Enter the first string:\n')
11:     s2 = input('Enter the second string:\n')
12:
13:     print(s1 is s2)
14:     print(s1 == s2)
15:
16: if __name__ == '__main__':
17:     main()
```

EqualityInt.java (Page 1 of 1)

```
 1: import java.util.Scanner;
 2:
 3: public class EqualityInt
 4: {
 5:    public static void main(String[] args)
 6:    {
 7:       Scanner scanner = new Scanner(System.in);
 8:
 9:       System.out.println("Enter the first int:");
10:       int i1 = scanner.nextInt();
11:
12:       System.out.println("Enter the second int:");
13:       int i2 = scanner.nextInt();
14:
15:       System.out.println(i1 == i2);
16:       // System.out.println(i1.equals(i2));
17:    }
18: }
```


equalityint.py (Page 1 of 1)

```
 1: #!/usr/bin/env python
 2:
 3: #-----------------------------------------------------------------------
 4: # equalityint.py
 5: # Author: Bob Dondero
 6: #-----------------------------------------------------------------------
 7:
 8: def main():
 9:
10:     i1 = int(input('Enter the first int:\n'))
11:     i2 = int(input('Enter the second int:\n'))
12:
13:     print(i1 is i2)
14:     print(i1 == i2)
15:
16: if __name__ == '__main__':
17:     main()
```

queue.py (Page 1 of 1)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# queue.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

class _Node:

    def __init__(self, item):
        self._item = item
        self._next = None
    def get_item(self):
        return self._item
    def get_next(self):
        return self._next
    def set_next(self, next):
        self._next = next

class Queue:

    def __init__(self):
        self._head_node = None
        self._tail_node = None

    def put(self, item):
        new_node = _Node(item)
        if self._tail_node is None:
            self._head_node = new_node
        else:
            self._tail_node.set_next(new_node)
        self._tail_node = new_node

    def get(self):
        if self._head_node is None:
            raise Exception('Empty queue')
        item = self._head_node.get_item()
        self._head_node = self._head_node.get_next()
        if self._head_node is None:
            self._tail_node = None
        return item

    def is_empty(self):
        return self._head_node is None
```

priorityqueue.py (Page 1 of 1)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# priorityqueue.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import queue

class PriorityQueue (queue.Queue):

    def put(self, item):
        new_node = queue._Node(item)

        if self._head_node is None:
            self._head_node = new_node
            self._tail_node = new_node
            return

        prev_node = None
        curr_node = self._head_node
        while curr_node is not None:
            if curr_node.get_item() < item:
                if prev_node is None:
                    self._head_node = new_node
                else:
                    prev_node.set_next(new_node)
                new_node.set_next(curr_node)
                return
            prev_node = curr_node
            curr_node = curr_node.get_next()

        self._tail_node.set_next(new_node)
        self._tail_node = new_node
```

priorityqueueclient.py (Page 1 of 1)

```
#!/usr/bin/env python

#--------------------------------------------------------------------------
# priorityqueueclient.py
# Author: Bob Dondero
#--------------------------------------------------------------------------

import sys
import priorityqueue

def main():

    pqueue = priorityqueue.PriorityQueue()

    print('Enter non-negative ints, one per line.')
    print('Enter a negative int to stop.')

    try:
        line = input()
        item = int(line)
        while item >= 0:
            pqueue.put(item)
            line = input()
            item = int(line)

        while not pqueue.is_empty():
            item = pqueue.get()
            print(item)

    except Exception as ex:
        print(ex, file=sys.stderr)
        sys.exit(1)

if __name__ == '__main__':
    main()
```

blank (Page 1 of 1)

This page is intentionally blank.

objectparam1.py (Page 1 of 1)

```
#!/usr/bin/env python

#-------------------------------------------------------------------------
# objectparam1.py
# Author: Bob Dondero
#-------------------------------------------------------------------------

class IntWrapper:
    def __init__(self, i):
        self._i = i
    def get(self):
        return self._i
    def set(self, i):
        self._i = i

#-------------------------------------------------------------------------

def my_func(iw2):
    iw2 = IntWrapper(6)

#-------------------------------------------------------------------------

def main():
    iw1 = IntWrapper(5)
    my_func(iw1)
    print(iw1.get())

#-------------------------------------------------------------------------

if __name__ == '__main__':
    main()
```

objectparam2.py (Page 1 of 1)

```
#!/usr/bin/env python

#-------------------------------------------------------------------------
# objectparam2.py
# Author: Bob Dondero
#-------------------------------------------------------------------------

class IntWrapper:
    def __init__(self, i):
        self._i = i
    def get(self):
        return self._i
    def set(self, i):
        self._i = i

#-------------------------------------------------------------------------

def my_func(iw2):
    iw2.set(6)

#-------------------------------------------------------------------------

def main():
    iw1 = IntWrapper(5)
    my_func(iw1)
    print(iw1.get())

#-------------------------------------------------------------------------

if __name__ == '__main__':
    main()

```