

## 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)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # fractionclient.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9: import fraction
10:
11: def main():
12:
13:     try:
14:
15:         line = input('Numerator 1: ')
16:         num1 = int(line)
17:         line = input('Denominator 1: ')
18:         den1 = int(line)
19:         line = input('Numerator 2: ')
20:         num2 = int(line)
21:         line = input('Denominator 2: ')
22:         den2 = int(line)
23:
24:         frac1 = fraction.Fraction(num1, den1)
25:         print('frac1:', str(frac1)) # Same as frac1.__str__()
26:
27:         frac2 = fraction.Fraction(num2, den2)
28:         print('frac2:', frac2) # print() calls str(frac2)
29:                                 # Same as frac2.__str__()
30:
31:         print('frac1 hashcode:', hash(frac1)) # Same as frac1.__hash__()
32:
33:         if frac1 == frac2: # Same as frac1.__eq__(frac2)
34:             print('frac1 equals frac2')
35:         if frac1 != frac2: # Same as frac1.__ne__(frac2)
36:             print('frac1 does not equal frac2')
37:         if frac1 < frac2: # Same as frac1.__lt__(frac2)
38:             print('frac1 is less than frac2')
39:         if frac1 > frac2: # Same as frac1.__gt__(frac2)
40:             print('frac1 is greater than frac2')
41:         if frac1 <= frac2: # Same as frac1.__le__(frac2)
42:             print('frac1 is less than or equal to frac2')
43:         if frac1 >= frac2: # Same as frac1.__ge__(frac2)
44:             print('frac1 is greater than or equal to frac2')
45:
46:         frac3 = -frac1 # Same as frac1.__neg__()
47:         print('-frac1:', frac3)
48:
49:         frac3 = frac1 + frac2 # Same as frac1.__add__(frac2)
50:         print('frac1 + frac2:', frac3)
51:
52:         frac3 = frac1 - frac2 # Same as frac1.__sub__(frac2)
53:         print('frac1 - frac2:', frac3)
54:
55:         frac3 = frac1 * frac2 # Same as frac1.__mul__(frac2)
56:         print('frac1 * frac2:', frac3)
57:
58:         frac3 = frac1 / frac2 # Same as frac1.__truediv__(frac2)
59:         print('frac1 / frac2:', frac3)
60:
61:     except Exception as ex:
62:         print(str(ex), file=sys.stderr)
63:
64: #-----
65:

```

## fractionclient.py (Page 2 of 2)

```

66: if __name__ == '__main__':
67:     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)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # queue.py
5: # Author: Bob Dondero
6: #-----
7:
8: class _Node:
9:
10:     def __init__(self, item):
11:         self._item = item
12:         self._next = None
13:     def get_item(self):
14:         return self._item
15:     def get_next(self):
16:         return self._next
17:     def set_next(self, next):
18:         self._next = next
19:
20: class Queue:
21:
22:     def __init__(self):
23:         self._head_node = None
24:         self._tail_node = None
25:
26:     def put(self, item):
27:         new_node = _Node(item)
28:         if self._tail_node is None:
29:             self._head_node = new_node
30:         else:
31:             self._tail_node.set_next(new_node)
32:         self._tail_node = new_node
33:
34:     def get(self):
35:         if self._head_node is None:
36:             raise Exception('Empty queue')
37:         item = self._head_node.get_item()
38:         self._head_node = self._head_node.get_next()
39:         if self._head_node is None:
40:             self._tail_node = None
41:         return item
42:
43:     def is_empty(self):
44:         return self._head_node is None

```

## priorityqueue.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # priorityqueue.py
5: # Author: Bob Dondero
6: #-----
7:
8: import queue
9:
10: class PriorityQueue (queue.Queue):
11:
12:     def put(self, item):
13:         new_node = queue._Node(item)
14:
15:         if self._head_node is None:
16:             self._head_node = new_node
17:             self._tail_node = new_node
18:         return
19:
20:         prev_node = None
21:         curr_node = self._head_node
22:         while curr_node is not None:
23:             if curr_node.get_item() < item:
24:                 if prev_node is None:
25:                     self._head_node = new_node
26:                 else:
27:                     prev_node.set_next(new_node)
28:                 new_node.set_next(curr_node)
29:                 return
30:             prev_node = curr_node
31:             curr_node = curr_node.get_next()
32:
33:         self._tail_node.set_next(new_node)
34:         self._tail_node = new_node

```

**priorityqueueclient.py (Page 1 of 1)**

```
1: #!/usr/bin/env python
2:
3: #-----
4: # priorityqueueclient.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9: import priorityqueue
10:
11: def main():
12:
13:     pqueue = priorityqueue.PriorityQueue()
14:
15:     print('Enter non-negative ints, one per line.')
16:     print('Enter a negative int to stop.')
17:
18:     try:
19:         line = input()
20:         item = int(line)
21:         while item >= 0:
22:             pqueue.put(item)
23:             line = input()
24:             item = int(line)
25:
26:         while not pqueue.is_empty():
27:             item = pqueue.get()
28:             print(item)
29:
30:     except Exception as ex:
31:         print(ex, file=sys.stderr)
32:         sys.exit(1)
33:
34: if __name__ == '__main__':
35:     main()
```

**blank (Page 1 of 1)**

```
1: This page is intentionally blank.
```

## objectparam1.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # objectparam1.py
5: # Author: Bob Dondero
6: #-----
7:
8: class IntWrapper:
9:     def __init__(self, i):
10:         self._i = i
11:     def get(self):
12:         return self._i
13:     def set(self, i):
14:         self._i = i
15:
16: #-----
17:
18: def my_func(iw2):
19:     iw2 = IntWrapper(6)
20:
21: #-----
22:
23: def main():
24:     iw1 = IntWrapper(5)
25:     my_func(iw1)
26:     print(iw1.get())
27:
28: #-----
29:
30: if __name__ == '__main__':
31:     main()

```

## objectparam2.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # objectparam2.py
5: # Author: Bob Dondero
6: #-----
7:
8: class IntWrapper:
9:     def __init__(self, i):
10:         self._i = i
11:     def get(self):
12:         return self._i
13:     def set(self, i):
14:         self._i = i
15:
16: #-----
17:
18: def my_func(iw2):
19:     iw2.set(6)
20:
21: #-----
22:
23: def main():
24:     iw1 = IntWrapper(5)
25:     my_func(iw1)
26:     print(iw1.get())
27:
28: #-----
29:
30: if __name__ == '__main__':
31:     main()
32:

```