euclid.py (Page 1 of 1)

```python
 1: #!/usr/bin/env python
 2:
 3: #-------------------------------------------------------------------
 4: # euclid.py
 5: # Author: Bob Dondero
 6: #-------------------------------------------------------------------
 7:
 8: def gcd(i, j):
 9:
10:     if (i == 0) and (j == 0):
11:         raise ZeroDivisionError(
12:             'gcd(i,j) is undefined if i and j are 0')
13:     i = abs(i)
14:     j = abs(j)
15:     while j != 0:   # Euclid's algorithm
16:         i, j = j, i%j
17:     return i
18:
19: #-------------------------------------------------------------------
20:
21: def lcm(i, j):
22:
23:     if (i == 0) or (j == 0):
24:         raise ZeroDivisionError(
25:             'lcm(i,j) is undefined if i or j is 0')
26:     i = abs(i)
27:     j = abs(j)
28:     return (i // gcd(i, j)) * j
```

euclidclient3.py (Page 1 of 1)

```python
 1: #!/usr/bin/env python
 2:
 3: #-------------------------------------------------------------
 4: # euclidclient3.py
 5: # Author: Bob Dondero
 6: #-------------------------------------------------------------
 7:
 8: import sys
 9: import euclid
10:
11: def main():
12:
13:     try:
14:         line = input('Enter the first integer: ')
15:         i = int(line)
16:
17:         line = input('Enter the second integer: ')
18:         j = int(line)
19:
20:         my_gcd = euclid.gcd(i, j)
21:         print('gcd:', my_gcd)
22:
23:         my_lcm = euclid.lcm(i, j)
24:         print('lcm:', my_lcm)
25:
26:     except ValueError:
27:         print('Error: Not an integer', file=sys.stderr)
28:         sys.exit(1)
29:     except EOFError:
30:         print('Error: Missing integer', file=sys.stderr)
31:         sys.exit(1)
32:     except ZeroDivisionError as ex:
33:         print(str(ex), file=sys.stderr)
34:         sys.exit(1)
35:
36: if __name__ == '__main__':
37:     main()
```

intmath/__init__.py (Page 1 of 1)

```
1: #!/usr/bin/env python
2:
```

intmath/euclid.py (Page 1 of 1)

```
 1: #!/usr/bin/env python
 2:
 3: #---------------------------------------------------------------------
 4: # euclid.py
 5: # Author: Bob Dondero
 6: #---------------------------------------------------------------------
 7:
 8: def gcd(i, j):
 9:
10:     if (i == 0) and (j == 0):
11:         raise ZeroDivisionError(
12:             'gcd(i,j) is undefined if i and j are 0')
13:
14:     i = abs(i)
15:     j = abs(j)
16:     while j != 0:  # Euclid's algorithm
17:         i, j = j, i%j
18:     return i
19:
20: #---------------------------------------------------------------------
21:
22: def lcm(i, j):
23:
24:     if (i == 0) or (j == 0):
25:         raise ZeroDivisionError(
26:             'lcm(i,j) is undefined if i or j is 0')
27:
28:     i = abs(i)
29:     j = abs(j)
30:     return (i // gcd(i, j)) * j
```

intmath/fibonacci.py (Page 1 of 1)

```python
 1: #!/usr/bin/env python
 2:
 3: #-----------------------------------------------------------------------
 4: # fibonacci.py
 5: # Author: Bob Dondero
 6: #-----------------------------------------------------------------------
 7:
 8: def fib(num):
 9:
10:     if num < 0:
11:         raise ValueError()
12:     if num == 0:
13:         return 0
14:     if num == 1:
15:         return 1
16:
17:     # The Wikipedia "Fibonacci Numbers" article describes some
18:     # approaches that are more efficient.
19:
20:     second_prev = 0
21:     first_prev = 1
22:     while num > 1:
23:         current = second_prev + first_prev
24:         second_prev = first_prev
25:         first_prev = current
26:         num -= 1
27:
28:     return current
```

blank (Page 1 of 1)

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

euclidclient4.py (Page 1 of 1)

```python
 1: #!/usr/bin/env python
 2:
 3: #-------------------------------------------------------------------
 4: # euclidclient4.py
 5: # Author: Bob Dondero
 6: #-------------------------------------------------------------------
 7:
 8: import sys
 9: import intmath.euclid
10:
11: def main():
12:
13:     try:
14:         line = input('Enter the first integer: ')
15:         i = int(line)
16:
17:         line = input('Enter the second integer: ')
18:         j = int(line)
19:
20:         my_gcd = intmath.euclid.gcd(i, j)
21:         print('gcd:', my_gcd)
22:
23:         my_lcm = intmath.euclid.lcm(i, j)
24:         print('lcm:', my_lcm)
25:
26:     except ValueError:
27:         print('Error: Not an integer', file=sys.stderr)
28:         sys.exit(1)
29:     except EOFError:
30:         print('Error: Missing integer', file=sys.stderr)
31:         sys.exit(1)
32:     except ZeroDivisionError as ex:
33:         print(str(ex), file=sys.stderr)
34:         sys.exit(1)
35:
36: if __name__ == '__main__':
37:     main()
```

fibclient.py (Page 1 of 1)

```python
 1: #!/usr/bin/env python
 2:
 3: #-------------------------------------------------------------------
 4: # fibclient.py
 5: # Author: Bob Dondero
 6: #-------------------------------------------------------------------
 7:
 8: import sys
 9: import intmath.fibonacci
10:
11: def main():
12:
13:     try:
14:         for i in range(10):
15:             print('fib(' + str(i) + ') = ' +
16:                 str(intmath.fibonacci.fib(i)))
17:
18:     except ValueError:
19:         print('Error: Not a positive integer', file=sys.stderr)
20:         sys.exit(1)
21:
22: if __name__ == '__main__':
23:     main()
```

fractionprelim.py (Page 1 of 2)

```python
 1: #!/usr/bin/env python
 2:
 3: #-----------------------------------------------------------------------
 4: # fractionprelim.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 to_string(self):
33:         if self._den == 1:
34:             return str(self._num)
35:         return '%d/%d' % (self._num, self._den)
36:
37:     def hash_code(self):
38:         return hash((self._num, self._den))    # Use tuple's __hash__().
39:
40:     def equals(self, other):
41:         return (self._num == other._num) and (self._den == other._den)
42:
43:     def compare_to(self, other):
44:         if (self._num * other._den) < (other._num * self._den):
45:             return -1
46:         if (self._num * other._den) > (other._num * self._den):
47:             return 1
48:         return 0
49:
50:     def negate(self):
51:         return Fraction(-self._num, self._den)
52:
53:     def add(self, other):
54:         new_num = (self._num * other._den) + (other._num * self._den)
55:         new_den = self._den * other._den
56:         return Fraction(new_num, new_den)
57:
58:     def subtract(self, other):
59:         new_num = (self._num * other._den) - (other._num * self._den)
60:         new_den = self._den * other._den
61:         return Fraction(new_num, new_den)
62:
63:     def multiply(self, other):
64:         new_num = self._num * other._num
65:         new_den = self._den * other._den
```

fractionprelim.py (Page 2 of 2)

```python
66:         return Fraction(new_num, new_den)
67:
68:     def divide(self, other):
69:         new_num = self._num * other._den
70:         new_den = self._den * other._num
71:         return Fraction(new_num, new_den)
```

fractionprelimclient.py (Page 1 of 2)

```python
 1: #!/usr/bin/env python
 2:
 3: #-----------------------------------------------------------------------
 4: # fraction1client.py
 5: # Author: Bob Dondero
 6: #-----------------------------------------------------------------------
 7:
 8: import sys
 9: import fractionprelim as fraction
10:
11: def main():
12:
13:     try:
14:         line = input('Numerator 1: ')
15:         num1 = int(line)
16:         line = input('Denominator 1: ')
17:         den1 = int(line)
18:         line = input('Numerator 2: ')
19:         num2 = int(line)
20:         line = input('Denominator 2: ')
21:         den2 = int(line)
22:
23:         frac1 = fraction.Fraction(num1, den1)
24:         print('frac1:', frac1.to_string())
25:
26:         frac2 = fraction.Fraction(num2, den2)
27:         print('frac2:', frac2.to_string())
28:
29:         print('frac1 hashcode:', frac1.hash_code())
30:
31:         if frac1.equals(frac2):
32:             print('frac1 equals frac2')
33:         if not frac1.equals(frac2):
34:             print('frac1 does not equal frac2')
35:
36:         comparison = frac1.compare_to(frac2)
37:         if comparison < 0:
38:             print('frac1 is less than frac2')
39:         if comparison > 0:
40:             print('frac1 is greater than frac2')
41:         if comparison <= 0:
42:             print('frac1 is less than or equal to frac2')
43:         if comparison >= 0:
44:             print('frac1 is greater than or equal to frac2')
45:
46:         frac3 = frac1.negate()
47:         print('-frac1:', frac3.to_string())
48:
49:         frac3 = frac1.add(frac2)
50:         print('frac1 + frac2:', frac3.to_string())
51:
52:         frac3 = frac1.subtract(frac2)
53:         print('frac1 - frac2:', frac3.to_string())
54:
55:         frac3 = frac1.multiply(frac2)
56:         print('frac1 * frac2:', frac3.to_string())
57:
58:         frac3 = frac1.divide(frac2)
59:         print('frac1 / frac2:', frac3.to_string())
60:
61:     except Exception as ex:
62:         print(str(ex), file=sys.stderr)
63:
64: #-----------------------------------------------------------------------
65:
```

fractionprelimclient.py (Page 2 of 2)

```python
66: if __name__ == '__main__':
67:     main()
```

euclidstrong.py (Page 1 of 1)

```python
 1: #!/usr/bin/env python
 2:
 3: #-----------------------------------------------------------------------
 4: # euclidstrong.py
 5: # Author: Bob Dondero
 6: #-----------------------------------------------------------------------
 7:
 8: def gcd(i, j):
 9:
10:     if not isinstance(i, int) or not isinstance(j, int):
11:         raise TypeError('gcd() arguments must be integers')
12:
13:     if (i == 0) and (j == 0):
14:         raise ZeroDivisionError(
15:             'gcd(i,j) is undefined if i and j are 0')
16:
17:     i = abs(i)
18:     j = abs(j)
19:     while j != 0:  # Euclid's algorithm
20:         i, j = j, i%j
21:     return i
22:
23: #-----------------------------------------------------------------------
24:
25: def lcm(i, j):
26:
27:     if not isinstance(i, int) or not isinstance(j, int):
28:         raise TypeError('lcm() arguments must be integers')
29:
30:     if (i == 0) or (j == 0):
31:         raise ZeroDivisionError(
32:             'lcm(i,j) is undefined if i or j is 0')
33:
34:     i = abs(i)
35:     j = abs(j)
36:     return (i // gcd(i, j)) * j
```