copybytes.py (Page 1 of 1)

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

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

import sys

MAX_BYTE_COUNT = 512  # Arbitrary

def main():

    if len(sys.argv) != 3:
        print('usage: python %s infile outfile' % sys.argv[0],
            file=sys.stderr)
        sys.exit(1)

    in_name = sys.argv[1]
    out_name = sys.argv[2]

    try:
        in_file = open(in_name, mode='rb')
        out_file = open(out_name, mode='wb')

        byte_array = in_file.read(MAX_BYTE_COUNT)
        while byte_array != b'':
            out_file.write(byte_array)
            byte_array = in_file.read(MAX_BYTE_COUNT)

        out_file.close()
        in_file.close()

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

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

copystrings.py (Page 1 of 1)

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

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

import sys

def main():

    if len(sys.argv) != 3:
        print('usage: python %s infile outfile' % sys.argv[0],
            file=sys.stderr)
        sys.exit(1)

    in_name = sys.argv[1]
    out_name = sys.argv[2]

    try:
        in_file = open(in_name, mode='r', encoding='utf-8')
        out_file = open(out_name, mode='w', encoding='utf-8')

        for line in in_file:
            out_file.write(line)

        out_file.close()
        in_file.close()

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

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

copystringsfinally.py (Page 1 of 1)

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

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

import sys

def main():

    if len(sys.argv) != 3:
        print('usage: python %s infile outfile' % sys.argv[0],
            file=sys.stderr)
        sys.exit(1)

    in_name = sys.argv[1]
    out_name = sys.argv[2]

    try:
        in_file = open(in_name, mode='r', encoding='utf-8')
        try:
            out_file = open(out_name, mode='w', encoding='utf-8')
            try:
                for line in in_file:
                    out_file.write(line)
            finally:
                out_file.close()
        finally:
            in_file.close()

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

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

copystringswith.py (Page 1 of 1)

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

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

import sys

def main():

    if len(sys.argv) != 3:
        print('usage: python %s infile outfile' % sys.argv[0],
            file=sys.stderr)
        sys.exit(1)

    in_name = sys.argv[1]
    out_name  = sys.argv[2]

    try:
        with open(in_name, mode='r', encoding='utf-8') as in_file:
            with open(out_name, mode='w', encoding='utf-8') as out_file:
                for line in in_file:
                    out_file.write(line)

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

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

linesort.py (Page 1 of 1)

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

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

import sys

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

def main():

    lines = []
    for line in sys.stdin:
        line = line.rstrip('\n')
        lines.append(line)  # or: lines += [line]
    lines.sort()
    for line in lines:
        print(line)

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

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

blank (Page 1 of 1)

```
This page is intentionally blank.
```

linesorttim.py (Page 1 of 2)

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

#-----------------------------------------------------------------------
# linesorttim.py
# Author:
# https://www.codespeedy.com/timsort-algorithm-implementation-in-python/
#-----------------------------------------------------------------------

import sys

minrun = 32

def InsSort(arr,start,end):
    for i in range(start+1,end+1):
        elem = arr[i]
        j = i-1
        while j>=start and elem<arr[j]:
            arr[j+1] = arr[j]
            j -= 1
        arr[j+1] = elem
    return arr

def merge(arr,start,mid,end):
    if mid==end:
        return arr
    first = arr[start:mid+1]
    last = arr[mid+1:end+1]
    len1 = mid-start+1
    len2 = end-mid
    ind1 = 0
    ind2 = 0
    ind  = start

    while ind1<len1 and ind2<len2:
        if first[ind1]<last[ind2]:
            arr[ind] = first[ind1]
            ind1 += 1
        else:
            arr[ind] = last[ind2]
            ind2 += 1
        ind += 1

    while ind1<len1:
        arr[ind] = first[ind1]
        ind1 += 1
        ind += 1

    while ind2<len2:
        arr[ind] = last[ind2]
        ind2 += 1
        ind += 1

    return arr

def TimSort(arr):
    n = len(arr)

    for start in range(0,n,minrun):
        end = min(start+minrun-1,n-1)
        arr = InsSort(arr,start,end)

    curr_size = minrun
    while curr_size<n:
        for start in range(0,n,curr_size*2):
            mid = min(n-1,start+curr_size-1)
```

linesorttim.py (Page 2 of 2)

```python
            end = min(n-1,mid+curr_size)
            arr = merge(arr,start,mid,end)
        curr_size *= 2
    return arr

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

def main():
    lines = []
    for line in sys.stdin:
        line = line.rstrip('\n')
        lines.append(line)  # or: lines += [line]
    TimSort(lines)
    for line in lines:
        print(line)

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

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

concord.py (Page 1 of 1)

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

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

import sys
import re

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

def process_line(line, concordance):

    line = line.lower()

    re_letters = re.compile(r'[a-z]+')
    words = re_letters.findall(line)

    for word in words:
        if word in concordance:
            concordance[word] += 1
        else:
            concordance[word] = 1

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

def main():

    concordance = {}

    for line in sys.stdin:
        process_line(line, concordance)

    #for word in concordance:
    #    print('%s: %d' % (word, concordance[word]))
    for word, count in concordance.items():
        print('%s: %d' % (word, count))

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

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

variadic.py (Page 1 of 1)

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

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

def my_func(i, j, *args, **kwargs):
    print(i)
    print(j)
    for arg in args:
        print(arg)
    for key, value in kwargs.items():
        print(key, value)

def main():
    my_func('a', 'b', 'c', 'd', key1='e', key2='f')

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

euclid.py (Page 1 of 1)

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

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

def gcd(i, j):

    if (i == 0) and (j == 0):
        raise ZeroDivisionError(
            'gcd(i,j) is undefined if i and j are 0')
    i = abs(i)
    j = abs(j)
    while j != 0:  # Euclid's algorithm
        i, j = j, i%j
    return i

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

def lcm(i, j):

    if (i == 0) or (j == 0):
        raise ZeroDivisionError(
            'lcm(i,j) is undefined if i or j is 0')
    i = abs(i)
    j = abs(j)
    return (i // gcd(i, j)) * j
```

euclidstrong.py (Page 1 of 1)

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

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

def gcd(i, j):

    if not isinstance(i, int) or not isinstance(j, int):
        raise TypeError('gcd() arguments must be integers')

    if (i == 0) and (j == 0):
        raise ZeroDivisionError(
            'gcd(i,j) is undefined if i and j are 0')

    i = abs(i)
    j = abs(j)
    while j != 0:  # Euclid's algorithm
        i, j = j, i%j
    return i

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

def lcm(i, j):

    if not isinstance(i, int) or not isinstance(j, int):
        raise TypeError('lcm() arguments must be integers')

    if (i == 0) or (j == 0):
        raise ZeroDivisionError(
            'lcm(i,j) is undefined if i or j is 0')

    i = abs(i)
    j = abs(j)
    return (i // gcd(i, j)) * j
```